ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΔΥΤΙΚΗΣ ΕΛΛΑΔΑΣ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΟΛΟΓΩΝ ΤΕ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ 1596

ΚΑΤΑΣΚΕΥΗ ΠΛΗΡΟΦΟΡΙΑΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΣΕ C++ ΓΙΑ ΑΥΤΟΜΑΤΟΠΟΙΗΣΗ ΛΕΙΤΟΥΡΓΙΩΝ ΕΤΑΙΡΙΑΣ

ΣΠΟΥΔΑΣΤΗΣ : ΧΡΟΝΗΣ ΚΩΝ/ΝΟΣ

ΕΠΟΠΤΕΥΩΝ ΚΑΘΗΓΗΤΗΣ ΚΑΡΕΛΗΣ ΔΗΜΗΤΡΙΟΣ

ΠΑΤΡΑ ΟΚΤΩΒΡΙΟΣ 2016

ΠΡΟΛΟΓΟΣ

Στην παρούσα πτυχιακή εργασία παρουσιάζουμε έναν οδηγό περιγραφής με τα βασικά χαρακτηριστικά και τις δυνατότητες της αντικειμενοστραφούς γλώσσας C++

Στο Α μέρος παρουσιάζουμε τη γλώσσα προγραμματισμού C++ η οποία αποτελεί επέκταση της C στο περιβάλλον του αντικειμενοστραφούς προγραμματισμού. Ο αντικειμενοστραφής προγραμματισμός στηρίζεται σε αρχές που έχουν τις ρίζες τους στη δεκαετία του '60 και συγκεκριμένα στη γλώσσα Simula. Η βασική φιλοσοφία του αντικειμενοστραφούς προγραμματισμού είναι να περιγράψει ένα σύστημα με φυσικό τρόπο. Για να επιτευχθεί αυτό, πρέπει να γίνει περιγραφή των αντικειμένων από τα οποία αποτελείται το σύστημα, να μελετηθεί η συμπεριφορά τους και ο τρόπος με τον οποίο αλληλεπιδρούν μεταξύ τους. Στην αντικειμενοστραφή προσέγγιση, το αντικείμενο είναι ένα προγραμματιστικό «πακέτο», το οποίο αποτελείται από διαδικασίες και δεδομένα σχετιζόμενα μεταξύ τους. Αυτές οι διαδικασίες ονομάζονται μέθοδοι ενώ τα δεδομένα ονομάζονται ιδιότητες. Οι μέθοδοι σε πολλές περιπτώσεις είναι γραμμένες σε κάποια διαδικαστική γλώσσα. Το πρόγραμμα, σε αυτήν την προσέγγιση, αποτελείται από αντικείμενα, τα οποία μπορούν να αλληλεπιδρούν μεταξύ τους με μηνύματα. Τα μηνύματα ενεργοποιούν τις μεθόδους των αντικειμένων, που με τη σειρά τους μπορεί να στείλουν άλλα μηνύματα.

Στο Β Μέρος παρουσιάζουμε ένα πληροφοριακό σύστημα που έχουμε κατασκευάσει σε C++ το οποίο αυτοματοποιεί όλες τις λειτουργίες μιας εταιρείας που έχει συναλλαγές με πελάτες και προμηθευτές και εμπορεύεται κάποια είδη. Έχουμε κατασκευάσει μια βάση δεδομένων η οποία περιλαμβάνει 3 βασικά αρχεία στα οποία καταχωρούνται οι πελάτες, οι προμηθευτές και όλα τα είδη της εταιρίας και στη συνέχεια καταγράφονται όλες οι συναλλαγές της εταιρίας όπως αγορές από προμηθευτές,, πληρωμές πελατών κ.λ.π.. Επίσης το πληροφοριακό σύστημα παρουσιάζει και στατιστικά στοιχεία όπως πλήθος πωλήσεων ανά πελάτη, συνολικές πληρωμές, μέσος όρος αγορών από προμηθευτές, πλήθος παραγγελιών από πελάτες κ.λ.π.

ΠΕΡΙΛΗΨΗ

Η πτυχιακή εργασία είναι δομημένη σε 2 βασικές ενότητες. Στην 1η ενότητα παρουσιάζουμε τη γλώσσα προγραμματισμού C++. Πιο συγκεκριμένα πρώτα περιγράφουμε τα βασικά χαρακτηριστικά της C (μεταβλητές, σταθερές και τύποι δεδομένων). Στη συνέχεια αναφέρουμε τις εντολές εισόδου και εξόδου της C και τις εντολές ελέγχου και επανάληψης παραθέτοντας κατάλληλα παραδείγματα σε κάθε εντολή. Μετά περιγράφουμε τους πίνακες, (μονοδιάστατους και δισδιάστατους) τους δείκτες και τη δυναμική δέσμευση μνήμης. Ακολούθως περιγράφουμε τις συναρτήσεις και τα ιδιαίτερα χαρακτηριστικά τους όπως για παράδειγμα τις εξορισμού παραμέτρους κα την υπερφόρτωση τους. Μετά αναφέρουμε ένα νέο χαρακτηριστικό της C++ που είναι οι αναφορές της και οι διαφορές της από τους δείκτες. Στη συνέχεια δίνουμε μεγάλο βάρος στις κλάσεις που χρησιμοποιεί η C++ και δίνουμε ιδιαίτερη έμφαση σε χαρακτηριστικά όπως δημιουργός και καταστροφέας. Περιγράφουμε όλους τους τύπους κληρονομικότητας και δίνουμε αρκετά παραδείγματα πάνω σε αυτή. Το τελευταίο χαρακτηριστικό της γλώσσας είναι η υπερφόρτωση τελεστών (αριθμητικών και τελεστών εισόδου και εξόδου) καθώς και τα προβλήματα που αυτοί παρουσιάζουν.

Στη 2η ενότητα περιγράφουμε ένα πληροφοριακό σύστημα που έχουμε κατασκευάσει στη γλώσσα C++ το οποίο αυτοματοποιεί όλες τις λειτουργίες μιας εμπορικής εταιρείας. Συγκεκριμένα έχουμε κατασκευάσει μια βάση δεδομένων σε C++ η οποία αποτελείται από 3 βασικά αρχεία: Το items.txt το οποίο περιλαμβάνει όλα τα είδη της εταιρείας, το customers.txt το οποίο περιλαμβάνει όλους τους πελάτες της εταιρίας και το suppliers.txt το οποίο περιλαμβάνει όλους τους προμηθευτές της εταιρίας. Στη συνέχεια καταγράφουμε όλες τις κινήσεις της εταιρίας όπως:

- Καταγραφή πωλήσεων σε πελάτες
- Καταγραφή πληρωμών από πελάτες
- Καταγραφή πληρωμών σε προμηθευτές
- Καταγραφή αγορών από προμηθευτές
- Εμφάνιση Υπολοίπου Πελατών και Προμηθευτών
- Στατιστικά Στοιχεία πελατών και προμηθευτών όπως μέσος όρος πωλήσεων, πλήθος αγορών κ.λ.π.

Περιεχόμενα

Α ΜΕΡΟΣ – ΓΛΩΣΣΑ C++

Η C++είναι μια γενικού σκοπού γλώσσα προγραμματισμού Η/Υ. Είναι μέσου επιπέδου γλώσσα, καθώς περιλαμβάνει έναν συνδυασμό χαρακτηριστικών από γλώσσες υψηλού και χαμηλού επιπέδου. Υποστηρίζει δομημένο, αντικειμενοστραφή και γενικό προγραμματισμό. Η γλώσσα αναπτύχθηκε από τον Bjarne Stroustruptο 1979 στα εργαστήρια Bellτης AT&T, ως βελτίωση της γλώσσας προγραμματισμού C και αρχικά ονομάστηκε "C with Classes", δηλαδή C με Κλάσεις. Μετονομάστηκε σε C++ το 1983. Οι βελτιώσεις ξεκίνησαν με την προσθήκη κλάσεων, και ακολούθησαν, μεταξύ άλλων, εικονικές συναρτήσεις, υπερφόρτωση τελεστών, πολλαπλή κληρονομικότητα, πρότυπα κ.α.

1 Λεξιλόγιο γλώσσας

Το λεξιλόγιο της C περιλαμβάνει τα ακόλουθα:

- δεσμευμένες λέξεις (reserved words)
- τελεστές (operators)
- αναγνωριστές (identifiers)

1.1 Δεσμευμένες λέξεις

Οι δεσμευμένες λέξεις μιας γλώσσας έχουν αυστηρά καθορισμένη έννοια και τρόπο χρήσης και χρησιμοποιούνται σε συγκεκριμένες θέσεις ενός προγράμματος. Μερικές από τις πιο χαρακτηριστικές δεσμευμένες λέξεις είναι οι εξής:

- ***int***, ***float***, ***do***, ***if***, ***for***, sizeof κ.λ.π.

1.2 Αναγνωριστές

Στην κατηγορία των αναγνωριστών ανήκουν λέξεις που κατασκευάζει ο προγραμματιστής. Τέτοιες λέξεις δίνει ο προγραμματιστής σε μεταβλητές, σταθερές. συναρτήσεις και δικούς του τύπους δεδομένων. Η δημιουργία ονομάτων στη C++ διέπεται από ένα σύνολο κανόνων οι οποίοι είναι ανεξάρτητοι από τον τύπου του αντικειμένου στον οποίο αναφέρεται το όνομα. Αυτοί οι κανόνες αναφέρονται στο κεφάλαιο

- το όνομα πρέπει να περιλαμβάνει το πολύ 31 χαρακτήρες

1.3 Μεταβλητές-Σταθερές

Μεταβλητή (variable) είναι μια ποσότητα η οποία μπορεί να μεταβάλλει την τιμή κατά τη διάρκεια εκτέλεσης ενός προγράμματος, ενώ σταθερά (constant) είναι μια ποσότητα η οποία έχει σταθερή τιμή σε όλη τη διάρκεια εκτέλεσης του προγράμματος. Μεταβλητές χρησιμοποιούνται σχεδόν σε κάθε πρόγραμμα (πλην ελαχίστων εξαιρέσεων) και είναι θέσεις μνήμης στις οποίες:

- εισάγουμε δεδομένα από το πληκτρολόγιο
- καταχωρούμε αποτελέσματα που υπολογίζουμε από το πρόγραμμα

Τα ονόματα των μεταβλητών επιλέγονται ελεύθερα από τον προγραμματιστή πρέπει όμως υποχρεωτικά να υπακούουν στους ακόλουθους κανόνες:

i. Ο αρχικός χαρακτήρας πρέπει να είναι γράμμα ή underscore (_)
ii. Οι υπόλοιποι χαρακτήρες μπορεί να είναι γράμματα, αριθμοί ή underscore(_)
iii. Το μέγιστο μέγεθος μιας μεταβλητής μπορεί να είναι 32 χαρακτήρες
iv. Παίζει ρόλο ο τρόπος γραφής πεζών-κεφαλαίων. Έτσι τα ονόματα rate, RATE και Rate είναι 3 διαφορετικές μεταβλητές
v. Δεν μπορούν να χρησιμοποιούνται δεσμευμένες λέξεις της γλώσσας ως μεταβλητές

Όλες οι μεταβλητές που χρησιμοποιούνται σε ένα πρόγραμμα πρέπει υποχρεωτικά να δηλώνονται πριν τη χρήση τους γιατί αλλιώς θα έχουμε συντακτικό λάθος. Η δήλωση μιας μεταβλητής γίνεται με την ακόλουθη εντολή:

τύπος όνομα μεταβλητής;

Παραδείγματα δηλώσεων

int x	η μεταβλητή x δηλώνεται ακέραιου τύπου
float y	η μεταβλητή y δηλώνεται πραγματικού τύπου
char z	η μεταβλητή z δηλώνεται τύπου χαρακτήρα
double k	η μεταβλητή k δηλώνεται τύπου ***double***

Όλες οι μεταβλητές σε ένα πρόγραμμα σε C++ πρέπει να δηλώνονται πριν από τη χρήση τους σε κάποιο τύπο δεδομένων. Ο τύπος δεδομένων καθορίζει το σύνολο τιμών από το οποίο μπορεί να πάρει τιμές μια μεταβλητή. Οι τύποι δεδομένων διακρίνονται σε απλούς ή ενσωματωμένους και σε σύνθετους.

1.4 Τύποι Δεδομένων στη C++

Οι τύποι δεδομένων δείχνουν το σύνολο τιμών από το οποίο μπορεί να πάρει τιμές μια μεταβλητή. Στη γλώσσα C++ υπάρχουν οι ακόλουθοι τύποι:

Τύπος	**Μνήμη**	**Περιγραφή**
int	2 bytes	Προσημασμένες ακέραιες τιμές από –32.768 έως +32.767
short ***int*** ή short	1 byte	Προσημασμένες ακέραιες τιμές από –128 έως +127
long ***int*** ή long	4 bytes	Προσημασμένες ακέραιες τιμές από –2.147.483.648 έως +2.147.483.647
unsigned ***int*** ή	2	Απρόσημες ακέραιες τιμές από 0 έως 65.535

unsigned	bytes	
char	1 byte	Το σύνολο των χαρακτήρων. χαρακτήρας αυτός μπαίνει μέσα σε αποστρόφους π.χ. 'A', '!' κ.λ.π.
signed ***char***	1 byte	Ένας χαρακτήρας με τιμές από –128 έως 127
unsigned ***char***	1 byte	Ένας χαρακτήρας με τιμές από 0 έως 255
float	4 bytes	Δεκαδικές τιμές από 3.4E-38 έως 3.4E+38 και –3.4E-38 έως -3.4E+38
double	8 bytes	Δεκαδικές τιμές από 1.7E-308 έως 1.7E+308 και –1.7E-308 έως -1.7E+308
bool	1 byte	Τιμές ***true***, ***false*** (μόνο στη C++)

1.5 Εντολές Εισόδου – Εξόδου

Με την εντολή εισόδου μπορούμε να εισάγουμε τιμές σε μεταβλητές. Έχει την ακόλουθη μορφή:

cin>>μεταβλητή; για να εισάγουμε μια τιμή σε μια μεταβλητή

ή

cin>>μεταβλητή1>>μεταβλητή2>>…; για να εισάγουμε πολλές τιμές σε διαφορετικές μεταβλητές

Με την εντολή εξόδου μπορούμε να εμφανίσουμε στην οθόνη τιμές μεταβλητών και σχόλια. Έχει την ακόλουθη μορφή:

cout<<”σχόλιο”; για εκτυπώσουμε ένα σχόλιο

ή

cout<<μεταβλητή; για να εκτυπώσουμε ένα αποτέλεσμα

ή

cout<<”σχόλιο”<<μεταβλητή; για εκτυπώσουμε ένα σχόλιο και ένα αποτέλεσμα μαζί

1.1 Εντολή #*include*

Γράφοντας την εντολή:

- #***include*** <filename> ψάχνει για το αρχείο που καθορίζουμε μόνο στον ειδικό κατάλογο για ***include*** files
- #***include*** "filename" ψάχνει για το αρχείο που καθορίζουμε και στον τρέχοντα κατάλογο αλλά και στο ***include*** directory

2 Εντολές Επιλογής

2.1 Απλή επιλογή

Στην απλή επιλογή ελέγχεται μια συνθήκη και εφόσον είναι αληθής τότε εκτελείται μια ομάδα εντολών 1, ενώ αν είναι ψευδής τότε εκτελείται μια ομάδα εντολών 2. Έχει την ακόλουθη σύνταξη:

```
if (συνθήκη)
{
   ομάδα εντολών 1;
}
else
{
     ομάδα εντολών 2;
}
```

2.2 Περιορισμένη επιλογή

Στην περιορισμένη επιλογή ελέγχεται μια συνθήκη και εφόσον είναι αληθής τότε εκτελείται μια ομάδα εντολών 1 ενώ αν είναι ψευδής τότε δεν εκτελείται τίποτα και το πρόγραμμα συνεχίζει απλώς στην επόμενη εντολή μετά το ***if***. Έχει την ακόλουθη σύνταξη:

```
if (συνθήκη)
{
   ομάδα εντολών 1;
}
```

2.3 Εμφωλευμένη επιλογή

Στην εμφωλευμένη επιλογή ελέγχεται μια συνθήκη 1 και εφόσον είναι αληθής τότε εκτελείται μια ομάδα εντολών 1. Αν είναι ψευδής τότε ελέγχεται μια συνθήκη 2 και εφόσον είναι αληθής τότε εκτελείται μια ομάδα εντολών 2. Αν είναι ψευδής τότε ελέγχεται μια συνθήκη 3 κ.ο.κ. Συνολικά ελέγχονται n συνθήκες και εκτελείται μόνο μια ομάδα εντολών. Αν δεν είναι καμία συνθήκη αληθής τότε εκτελείται μια ομάδα εντολών n+1. Έχει την ακόλουθη σύνταξη:

```
if (συνθήκη 1)
{
   ομάδα εντολών 1;
}
else
      if (συνθήκη 2)
```

```
{
   ομάδα εντολών 2;
}
else
……..
     if (συνθήκη n)
     {
        ομάδα εντολών n;
     }
     else
          {
            ομάδα εντολών n+1;
          }
```

Σημειώσεις

1.Σε κάθε εντολή επιλογής η συνθήκη είναι μια λογική παράσταση η οποία παίρνει είτε την τιμή αλήθεια (***true***) είτε την τιμή ψέμα (***false***). Το 0 αντιστοιχεί στη λογική τιμή αλήθεια και ένας αριθμός ¹ 0 αντιστοιχεί στην λογική τιμή ψέμα.

2.Κάθε ομάδα εντολών μπορεί να είναι είτε μια μόνο εντολή ή ένα μπλοκ από εντολές μέσα σε άγκιστρα. Αν είναι μόνο μια εντολή τότε μπορούμε να παραλείψουμε τα άγκιστρα.

3 Εντολές Επανάληψης

Οι εντολές επανάληψης (ανακύκλωσης) προκαλούν την εκτέλεση μιας εντολής ή μιας ομάδας εντολών μέσα σε άγκιστρα είτε ένα προκαθορισμένο αριθμό επαναλήψεων είτε για όσο χρόνο μια συνθήκη είναι αληθής. Στη C++ χρησιμοποιούμε 3 εντολές επανάληψης.

3.1 Εντολή επανάληψης *for*

Η εντολή ***for*** επαναλαμβάνει μια ομάδα εντολών συνήθως ένα συγκεκριμένο αριθμό φορών αν και μπορεί να χρησιμοποιηθεί ακόμα και για την επανάληψη μιας ομάδας εντολών για όσο χρόνο μια συνθήκη είναι αληθής. Έχει την ακόλουθη σύνταξη:

for (αρχικές τιμές; συνθήκη; μεταβολές)

{

ομάδα εντολών;

}

Οι αρχικές τιμές μπορεί να είναι μια ή περισσότερες εντολές που διαχωρίζονται με κόμμα και έχουν στόχο να εκτελεστούν κάποιες εντολές πριν αρχίσει η επανάληψη. Η συνθήκη μπορεί να είναι απλή είτε να συγκροτείται από επιμέρους συνθήκες χωριζόμενες με κόμμα (σύνθετη) και για όσο χρόνο είναι αληθής εκτελείται η ομάδα εντολών ανάμεσα στα άγκιστρα. Μετά από κάθε επανάληψη εκτελούνται οι εντολές που βρίσκονται στο τμήμα μεταβολές οι οποίες έχουν σαν σκοπό να μεταβάλλουν την τιμή της συνθήκης. Στη συνέχεια ελέγχεται πάλι η νέα τιμή της συνθήκης και εφόσον εξακολουθεί να είναι αληθής επαναλαμβάνεται η εκτέλεση της ομάδας εντολών. Όταν η συνθήκη γίνει ψευδής τότε η εντολή ***for*** τερματίζεται και το πρόγραμμα συνεχίζει στην επόμενη εντολή μετά την ***for***.

3.2 Εντολή Επανάληψης *while*

Η εντολή ***while*** χρησιμοποιείται όπως και η ***for*** για να επαναλάβουμε την εκτέλεση μιας εντολής ή μιας ομάδας εντολών μέσα σε άγκιστρα είτε ένα συγκεκριμένο αριθμό φορών είτε για όσο χρόνο μια συνθήκη είναι αληθής. Πρώτα ελέγχεται η συνθήκη και εφόσον είναι αληθής τότε εκτελείται η ομάδα εντολών. Για όσο χρόνο η συνθήκη (απλή ή σύνθετη) είναι αληθής (***true***) τότε επαναλαμβάνεται η εκτέλεση της ομάδας εντολών. Όταν η συνθήκη γίνει ψευδής η εντολή ***while*** τερματίζεται και το πρόγραμμα συνεχίζει στην επόμενη εντολή μετά την ***while***. Έχει την ακόλουθη σύνταξη:

while (συνθήκη)

{

ομάδα εντολών;

}

3.3 Εντολή επανάληψης *do..while*

Η εντολή ***do while*** είναι αντίστοιχη με την εντολή ***while*** με τη διαφορά ότι πρώτα εκτελείται η ομάδα εντολών και μετά ελέγχεται η συνθήκη, δηλαδή η ομάδα εντολών θα εκτελεστεί τουλάχιστον μια φορά ακόμα και αν η συνθήκη είναι ψευδής. Έχει την ακόλουθη σύνταξη:

do

{

ομάδα εντολών;

}***while*** (συνθήκη);

4 Πίνακες

Οι πίνακες είναι διαδοχικές θέσεις μνήμης στις οποίες τοποθετούνται δεδομένα του ιδίου τύπου. Στα δεδομένα ενός πίνακα αναφερόμαστε με ένα δείκτη θέσης (μονοδιάστατος πίνακας) ή με ένα ζεύγος δεικτών θέσης (δισδιάστατος πίνακας).

4.1 Μονοδιάστατοι πίνακες

Τα στοιχεία του πίνακα είναι όπως αναφέραμε του ιδίου τύπου και τοποθετούνται σε διαδοχικές θέσεις μνήμης. Λέγεται μονοδιάστατος ή γραμμικός πίνακας γιατί όλα τα στοιχεία του είναι τοποθετημένα σε μια γραμμή όπως φαίνεται και στο ακόλουθο σχήμα:

0	**1**	**2**	**3**	**4**	**5**
12	-7	45	7	4	8

Με τα έντονα γράμματα φαίνονται οι θέσεις του πίνακα. Παρατηρούμε ότι οι θέσεις ενός μονοδιάστατου πίνακα αριθμούνται αρχίζοντας από το 0.

4.1.1 Δήλωση μονοδιάστατου πίνακα

Η δήλωση ενός μονοδιάστατου πίνακα γίνεται με την ακόλουθη εντολή:

τύπος όνομα_πίνακα [πλήθος στοιχείων];

Για παράδειγμα γράφοντας σε ένα πρόγραμμα τη δήλωση ***int*** grades[5]; ορίζουμε ένα πίνακα με όνομα grades που συγκροτείται από 5 ακεραίους σε διαδοχικές θέσεις μνήμης.

grades[0]	grades[1]	grades[2]	grades[3]	grades[4]

Γενικά το στοιχείο ενός μονοδιάστατου πίνακα στη θέση i συμβολίζεται σαν Πίνακας[i]. Κάθε μία από τις διαδοχικές θέσεις μνήμης που συγκροτούν ένα πίνακα αποτελούν ένα στοιχείο του πίνακα.

grades[0] Ο ακέραιος που υπάρχει στην πρώτη θέση του πίνακα grades (1ο στοιχείο)

grades [1] Ο ακέραιος που υπάρχει στην δεύτερη θέση του πίνακα grades (2ο στοιχείο)

grades [2] Ο ακέραιος που υπάρχει στην τρίτη θέση του πίνακα grades (3ο στοιχείο)

grades [3] Ο ακέραιος που υπάρχει στην τέταρτη θέση του πίνακα grades (4ο στοιχείο)

grades [4] Ο ακέραιος που υπάρχει στην πέμπτη θέση του πίνακα grades (5ο στοιχείο)

4.1.2 Αρχικές τιμές μονοδιάστατου πίνακα

Μαζί με την δήλωση ενός πίνακα μπορούμε να δώσουμε και αρχικές τιμές στα στοιχεία του πίνακα με εντολή της μορφής:

int num[5]= {10, 25, 15, 35,5};

Τότε ο πίνακας num θα έχει τα ακόλουθα στοιχεία:

10	25	15	35	5

Δηλαδή θα ισχύουν: num[0]=10, num[1]=25, num[2]=15, num[3]=35, num[4]=5. Δηλώνοντας κάποιο πίνακα χωρίς αρχικές τιμές τα στοιχεία του έχουν τιμές απροσδιόριστες.

4.1.3 Εκχώρηση τιμών στα στοιχεία μονοδιάστατου πίνακα

Από τη στιγμή που έχουμε δηλώσει έναν πίνακα (με αρχικές τιμές ή χωρίς αρχικές τιμές) δεν υπάρχει τρόπος να εκχωρήσουμε, με μια εντολή, τιμές σε όλα τα στοιχεία του. Δηλαδή είναι λάθος να γράψουμε: num ={4, 8 12, 5, 20};

Οι τιμές καταχωρούνται μόνο σε μεμονωμένα στοιχεία του πίνακα κάθε φορά. Για παράδειγμα δίνοντας τις εντολές: num[0]=4; num[1]=8; num[2]=12; num[3]=5; num[4]=20; Ο πίνακας num θα έχει τα ακόλουθα στοιχεία:

4	8	12	5	20

4.1.4 Διάβασμα μονοδιάστατου πίνακα

Για να διαβάσουμε (γεμίσουμε) ένα μονοδιάστατο πίνακα με τιμές γράφουμε τις ακόλουθες εντολές: (έστω ότι έχει προηγηθεί η δήλωση ***float*** grades[5];)

```
for (i=0;i<5;i++)
{
   cout<<"Give value";
   cin>>grades[i];
}
```

4.1.5 Εκτύπωση μονοδιάστατου πίνακα

Για να εκτυπώσουμε (εμφανίσουμε) τα στοιχεία ενός μονοδιάστατου πίνακα στην οθόνη γράφουμε τις ακόλουθες εντολές: (έστω ότι έχει προηγηθεί η δήλωση ***float*** grades[5] και βέβαια ο πίνακας grades έχει πάρει τιμές)

```
for (i=0;i<5;i++)
    cout<<grades[i];
```

4.1.6 Μέγεθος μονοδιάστατου πίνακα

Αν έχουμε ορίσει ένα πίνακα τότε το πλήθος όλων των bytes της μνήμης που αφιερώνονται για τα στοιχεία του πίνακα προσδιορίζεται μέσω του τελεστή sizeof. Για παράδειγμα μετά την δήλωση ***int*** num[]={1,2,3,4,5}; η εντολή ***cout***<<"Size of array num in

bytes is "<<sizeof(num)<<***endl***; θα μας δώσει εκτυπώσει 10 (5 ακέραιοι επί 2 bytes ο καθένας).

4.2 Δισδιάστατοι πίνακες

Τα στοιχεία του πίνακα είναι και πάλι του ιδίου τύπου και βρίσκονται επίσης σε διαδοχικές θέσεις μνήμης. Λέγεται δισδιάστατος ή ορθογώνιος πίνακας γιατί όλα τα στοιχεία του είναι τοποθετημένα σε γραμμές και στήλες όπως φαίνεται και στο ακόλουθο σχήμα:

	0	**1**	**2**	**3**	**4**
0	12	-7	45	7	4
1	14	8	10	6	60
2	-9	5	20	3	2

Με τα έντονα γράμματα φαίνονται πάλι οι θέσεις του πίνακα. Παρατηρούμε ότι οι δείκτες και των γραμμών και των στηλών αριθμούνται από το 0.

4.2.1 Δήλωση δισδιάστατου πίνακα

Η δήλωση ενός μονοδιάστατου πίνακα γίνεται με την ακόλουθη εντολή:

τύπος όνομα_πίνακα [πλήθος γραμμών][πλήθος στηλών];

Με τη δήλωση αφιερώνεται χώρος μνήμης σε διαδοχικές θέσεις κατάλληλος για να δεχθεί δεδομένα του περιγραφόμενου τύπου και σε πλήθος όσο το γινόμενο των διαφορετικών πληθών των δεικτών. Για παράδειγμα γράφοντας σε ένα πρόγραμμα τη δήλωση ***int*** grades[3][5]; ορίζουμε ένα πίνακα με όνομα grades που αποτελείται από 3 γραμμές και 5 στήλες και συνολικά από 15 στοιχεία τύπου ***int***:

grades[0, 0]	grades[0, 1]	grades[0, 2]	grades[0, 3]	grades[0, 4]
grades[1, 0]	grades[1, 1]	grades[1, 2]	grades[1, 3]	grades[1, 4]
grades[2, 0]	grades[2, 1]	grades[2, 2]	grades[2, 3]	grades[2, 4]

Γενικά το στοιχείο ενός δισδιάστατου πίνακα στη γραμμή i και στη στήλη j συμβολίζεται σαν Πίνακας[i,j].

4.2.2 Αρχικές τιμές δισδιάστατου πίνακα

Μαζί με την δήλωση ενός δισδιάστατου πίνακα μπορούμε να δώσουμε και αρχικές τιμές στα στοιχεία του πίνακα με εντολή της μορφής:

int grades[2][5]={100,90,80,70,60,50,40,30,20,10};

Τότε στον πίνακα grades οι ακέραιοι μέσα στις αγκύλες τοποθετούνται κατά γραμμές επομένως θα έχουμε τα στοιχεία:

100	90	80	70	60
50	40	30	20	10

Δηλαδή το grades[0][0] θα είναι ο ακέραιος 100, το grades[1] [2] θα είναι ακέραιος 30 κ.λ.π. Η προηγούμενη δήλωση θα μπορούσε να γίνει και ως εξής:

int grades[2][5]= {{100,90,80,70,60}, {50,40,30,20,10}};

4.2.3 Εκχώρηση τιμών στα στοιχεία δισδιάστατου πίνακα

Από τη στιγμή που θα δηλώσουμε ένα πίνακα δεν υπάρχει τρόπος να δώσουμε με μια εντολή νέες τιμές σε όλα τα στοιχεία του (όπως συμβαίνει κατά την δήλωσή του με αρχικές τιμές). Δηλαδή μετά από τη δήλωση: ***int*** grades[2][5]; είναι λάθος να γράψουμε:

grades={{100,90,80,70,60,50,40,30,20,10}};

Μπορούμε όμως να δίνουμε τιμές σε μεμονωμένα στοιχεία του πίνακα γράφοντας:

grades[1][3]=100; grades[0][2]=500;

5 Συναρτήσεις

5.1 Εξορισμού (Default) Παράμετροι συνάρτησης (Καθιερωμένες παράμετροι συνάρτησης)

Η C++ μας δίνει τη δυνατότητα (που δεν υπάρχει στη C) να ορίζουμε αρχικές τιμές σε παραμέτρους με δήλωση της μορφής:

τύπος όνομα_τυπικής_παραμέτρου=τιμή

Έχοντας ορίσει σε μια συνάρτηση παραμέτρους με αρχικές τιμές λέμε ότι η συνάρτηση έχει καθιερωμένες (default) παραμέτρους. Οι καθιερωμένες παράμετροι τοποθετούνται σαν τελευταίες παράμετροι της συνάρτησης. Η συνάρτηση μπορεί να κληθεί με όλες ή με παράλειψη μερικών από τις τελευταίες καθιερωμένες παραμέτρους της. Οι παραλειπόμενες παράμετροι κατά την κλήση θα έχουν ως τιμή την αρχική τιμή που προσδιορίστηκε (την καθιερωμένη). Με τον τρόπο αυτό μπορούμε να καλούμε μια συνάρτηση με διαφορετικό πλήθος παραμέτρων από κλήση σε κλήση.

Σημείωση

Η αρχικοποίηση μιας default παραμέτρου μπορεί να γίνει με οποιαδήποτε τιμή και όχι υποχρεωτικά με μηδέν

<u>Παράδειγμα 1</u>

Στο παράδειγμα αυτό ορίζουμε μια συνάρτηση με το όνομα sum με default παραμέτρους

#***include*** <***iostream.h***>

double sum(***int*** k=0, ***int*** m=0) //Η συνάρτηση sum μπορεί να κληθεί με 0, 1 ή 2 παραμέτρους

```
{
    cout<<"A: ";
    return k+m;
}

double sum(int k, int m, int n) //Η συνάρτηση sum μπορεί να κληθεί μόνο με 3 παραμέτρους
{
    cout<<"B: ";
    return k+m+n;
}

void main()
{
    cout<<"sum() = "<<sum()<<endl; //καλείται η πρώτη συνάρτηση sum()
    cout<<"sum(10) = "<<sum(10)<<endl; //καλείται η πρώτη συνάρτηση sum()
    cout<<"sum(10,20) = "<<sum(10,20)<<endl; //καλείται η πρώτη συνάρτηση sum()
    cout<<"sum(10,20,30) = "<<sum(10,20,30)<<endl; //καλείται η δεύτερη συνάρτηση sum()
}
```

Αποτελέσματα Προγράμματος

```
A: sum()=0
A: sum(10)=10
A: sum(10,20)=30
B: sum(10,20,30)=60
```

Βασική Παρατήρηση

Η υπερφόρτωση δεν θα μπορούσε να γίνει με την τρίτη παράμετρο να έχει default τιμή, διότι καλώντας τη συνάρτηση sum με 2 παραμέτρους δεν θα ήταν δυνατό να καθοριστεί ποια από τις 2 συναρτήσεις πρέπει να εκτελεστεί. Συγκεκριμένα αν επιχειρούσαμε να το κάνουμε αυτό θα προέκυπτε συντακτικό σφάλμα.

Παράδειγμα 2

Στο παράδειγμα αυτό ορίζουμε τη συνάρτηση με το όνομα showchar με default παραμέτρους οι οποίες διαφέρουν μεταξύ τους μόνο ως προς τη σειρά που γράφονται. Ακόμα και έτσι το κύριο πρόγραμμα γνωρίζει ποια συνάρτηση να καλέσει

```
#include <iostream.h>

void showchar(int n, char a) //Η συνάρτηση showchar μπορεί να κληθεί κατά σειρά πρώτα με ακέραια παράμετρο και μετά με παράμετρο χαρακτήρα
{
    for (int i=0; i<n; i++)
        cout<<a;
}

void showchar(char a, int n) //Η συνάρτηση showchar μπορεί να κληθεί κατά σειρά πρώτα με παράμετρο χαρακτήρα και μετά με ακέραια παράμετρο
{
    for (int i=0; i<n; i++)
        cout<<a;
```

```
}

void main()
{
    showchar(3,'A'); //κλήση με ακέραιο, χαρακτήρα (καλείται η πρώτη συνάρτηση showchar)
    cout<<endl;
    showchar('A',3); //κλήση με χαρακτήρα, ακέραιο (καλείται η δεύτερη συνάρτηση showchar)
}
```

Βασική Παρατήρηση
Όταν καλούμε συνάρτηση και οι πραγματικές παράμετροι δεν έχουν τον τύπο των τυπικών παραμέτρων τότε γίνεται μετατροπή (αν είναι επιτρεπτή) της πραγματικής τιμής σε τιμή που έχει τον τύπο της τυπικής παραμέτρου. Στη C++ κάθε ακέραιος επιτρέπεται να μετατραπεί σε τύπου χαρακτήρα και το αντίστροφο. Η περίπτωση που θα χρειαστούν μετατροπές στις πραγματικές παραμέτρους μπορεί να προκαλέσει σφάλμα. Για παράδειγμα ας υποθέσουμε ότι καλούμε τη συνάρτηση showchar με την εντολή showchar(10,10); Τότε δίνουμε και τις 2 παραμέτρους ακέραιου τύπου. Δεν έχουμε όμως ορίσει τέτοια υπερφόρτωση. Θα επιχειρηθεί να γίνει μετατροπή των παραμέτρων ώστε να ταιριάζουν με τον τύπο των τυπικών παραμέτρων των συναρτήσεων που έχουν οριστεί. Τέτοιο ταίριασμα όμως μπορεί να γίνει και στις 2 συναρτήσεις (αφού είπαμε ότι ο τύπος char μπορεί να μετατραπεί σε int και το αντίστροφο). Στην περίπτωση αυτή δεν είναι ξεκάθαρο ποια από τις 2 συναρτήσεις πρέπει να κληθεί και ο compiler θα δώσει σφάλμα αμφιβολίας (ambiguity error).

Παράδειγμα 3

```
#include <iostream.h>

double sum (double a=0, double b=0, double c=0, double d=0) //όλες οι παράμετροι έχουν default τιμές μηδέν
{
   return a+b+c+d;
}

void main()
{
   cout<<"sum() = "<<sum()<<endl;
   cout<<"sum(100.5) = "<<sum(100.5)<<endl;
   cout<<"sum(10,20) = "<<sum(10,20)<<endl;
   cout<<"sum(10,20,-5.5) = "<<sum(10,20,-5.5)<<endl;
   cout<<"sum(1,2,3,4) = "<<sum(1,2,3,4)<<endl;
}
```

5.2 Υπερφόρτωση Συναρτήσεων

Ένα από τα πλέον δυναμικά χαρακτηριστικά της C++ είναι η δυνατότητα να χρησιμοποιούνται διαφορετικές συναρτήσεις με το ίδιο όνομα. Στη C++ μια συνάρτηση αναγνωρίζεται όχι μόνο από το όνομα της αλλά και από το πλήθος, τον τύπο και τη σειρά των παραμέτρων της. Δύο ή περισσότερες συναρτήσεις μπορεί να μοιράζονται το ίδιο όνομα

σε ένα πρόγραμμα. Ορίζοντας μια νέα συνάρτηση με όνομα παλιάς λέμε ότι δημιουργούμε **υπερφόρτωση (overloading) της παλιάς συνάρτησης**. Αυτό γίνεται όταν θέλουμε η ίδια συνάρτηση να εκτελεί **διαφορετικές λειτουργίες ανάλογα με τα ορίσματα που λαμβάνει**. Αν σε ένα πρόγραμμα έχουμε υπερφορτώσεις μιας συνάρτησης και θέλουμε να καλέσουμε μια από αυτές τότε θα πρέπει να δώσουμε μετά το όνομα της πραγματικές παραμέτρους που αντιστοιχίζονται (ως προς το πλήθος, τύπο και τη σειρά) με τις τυπικές παραμέτρους.

6 Συμβολοσειρές

Μία συμβολοσειρά (***string***) είναι μια ομάδα χαρακτήρων μέσα σε ένα ζεύγος διπλών εισαγωγικών π.χ. "Holidays", "Programming" κ.λ.π. Σε ένα πρόγραμμα χρησιμοποιούμε συνήθως τις συμβολοσειρές για την εκτύπωση μηνυμάτων στην οθόνη π.χ. η εντολή printf("Give a number"); εκτυπώνει τη φράση Give a number στην οθόνη του Η/Υ. Η συμβολοσειρά τοποθετείται στη μνήμη του Η/Υ σαν ένας πίνακας χαρακτήρων με ένα επιπλέον χαρακτήρα στο τέλος του. Αυτός ο χαρακτήρας ονομάζεται μηδενικός (null) και συμβολίζεται ως '\0'. Π.χ. το ***string*** Holidays τοποθετείται στη μνήμη ως εξής:

H	o	l	I	D	a	Y	s	\0

6.1 Ορισμός πίνακα συμβολοσειράς

Μια συμβολοσειρά μπορεί να τοποθετηθεί σε ένα πίνακα τύπου ***char*** δίνοντας αρχική τιμή π.χ. ***char*** s[]="one ***string***"; (προσαρμοζόμενη διάσταση) ή με τη δήλωση ***char*** s[11]="one ***string***"; (δεδομένη διάσταση). Αν και οι χαρακτήρες που τοποθετούνται μέσα στις αποστρόφους είναι 10 και στις 2 προηγούμενες δηλώσεις δημιουργείται ένας μονοδιάστατος πίνακας χαρακτήρων με όνομα s που έχει 11 στοιχεία (στο τέλος του πίνακα τοποθετείται αυτόματα ο μηδενικός χαρακτήρας). Το μέγιστο πλήθος στοιχείων του πίνακα χαρακτήρων στον οποίο θα τοποθετηθεί μια συμβολοσειρά πρέπει να είναι μεγαλύτερο τουλάχιστον κατά ένα από το πλήθος χαρακτήρων που θέλουμε να περιλαμβάνονται στη συμβολοσειρά προκειμένου να τοποθετηθεί στο τέλος του πίνακα ο μηδενικός χαρακτήρας.

6.2 Πίνακες με στοιχεία συμβολοσειρές

Ένας μονοδιάστατος πίνακας που τα στοιχεία του είναι συμβολοσειρές στην πραγματικότητα είναι ένας πίνακας 2 διαστάσεων τύπου ***char*** αφού το κάθε στοιχείο του (που είναι μια συμβολοσειρά) είναι ένας μονοδιάστατος πίνακας τύπου ***char***. Έτσι τους μονοδιάστατους πίνακες με συμβολοσειρές τους χειριζόμαστε όπως τους δισδιάστατους πίνακες.

6.3 Δήλωση πίνακα με αρχικές τιμές συμβολοσειρές

Με την δήλωση ***char*** s[ ][15] = {"ΤΕΙ Πειραιά", "ΤΕΙ Αθηνών", "ΤΕΙ Χαλκίδας"}; δημιουργείται ο ακόλουθος δισδιάστατος πίνακας s που έχει μορφή:

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
0	Τ	Ε	Ι		Π	Ε	Ι	ρ	α	Ι	ά	\0			

1	T	E	I		A	Θ	H	ν	ώ	ν	\0				
2	T	E	I		X	A	λ	κ	ί	δ	α	ς	\0		

Τα στοιχεία του πίνακα μπορούμε να τα προσπελάσουμε με δύο δείκτες που συμβολίζουν τη γραμμή και τη στήλη αντίστοιχα π.χ. το στοιχείο s[0][1] είναι ο χαρακτήρας 'Ε', το στοιχείο s[2][11] είναι ο χαρακτήρας 'ς' κ.λ.π.

7 Δείκτες

Η μεταβλητή ενός δείκτη (pointer) έχει σαν περιεχόμενο τη διεύθυνση μνήμης (σε δεκαεξαδική μορφή) μιας άλλης μεταβλητής (αυτής στην οποία δείχνει).

7.1 Δήλωση μεταβλητής δείκτη

Για να δηλώσουμε μια μεταβλητή δείκτη γράφουμε την ακόλουθη εντολή:

τύπος *όνομα_μεταβλητής;

π.χ. γράφοντας την εντολή ***int*** *x; δηλαδή η x είναι δείκτης (μεταβλητή δείκτης) που δείχνει σε ακέραιο (δηλαδή σε μια θέση μνήμης με ακέραιο περιεχόμενο). Σχηματικά έχουμε x **à** ***int***

π.χ. γράφοντας την εντολή ***float*** *y; δηλαδή η y είναι δείκτης (μεταβλητή δείκτης) που δείχνει σε πραγματικό (δηλαδή σε μια θέση μνήμης με πραγματικό περιεχόμενο). Σχηματικά έχουμε y **à** ***float***

π.χ. γράφοντας την εντολή ***char*** *z; δηλαδή η z είναι δείκτης (μεταβλητή δείκτης) που δείχνει σε χαρακτήρα (δηλαδή σε μια θέση μνήμης με περιεχόμενο χαρακτήρα). Σχηματικά έχουμε z **à** ***char***

7.2 Απόδοση τιμής σε δείκτη

Για να καταχωρίσουμε τιμή σε μια μεταβλητή δείκτη γράφουμε την εντολή:

μεταβλητή_δείκτης= &μεταβλητή που δείχνει

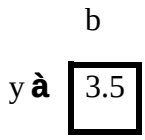
Το σύμβολο & υποδηλώνει διεύθυνση.

Για παράδειγμα αν έχουμε τις δηλώσεις ***int*** a=5, *x; τότε με την εντολή x=&a; καταχωρούμε στο δείκτη x ως περιεχόμενο τη διεύθυνση της μεταβλητής a. Σχηματικά έχουμε:

a

x **à** 5

Επίσης αν έχουμε τις δηλώσεις ***float*** b=3.5, *y; τότε με την εντολή y=&b; καταχωρούμε στο δείκτη y ως περιεχόμενο τη διεύθυνση της μεταβλητής b. Σχηματικά έχουμε:



7.3 Προσπέλαση μεταβλητής στην οποία δείχνει ο δείκτης

Για να προσπελάσουμε το περιεχόμενο της μεταβλητής στην οποία δείχνει ένας δείκτης χρησιμοποιούμε την εντολή:

*μεταβλητή_δείκτης

Για παράδειγμα αν έχουμε τις δηλώσεις ***int*** a=5, *x; τότε με την εντολή x=&a; καταχωρούμε στο δείκτη x ως περιεχόμενο τη διεύθυνση της μεταβλητής a και με την εντολή ***cout***<<*x; τυπώνεται η τιμή 5.

Επίσης αν έχουμε τις δηλώσεις ***float*** b=3.5, *y; τότε με την εντολή y=&b; καταχωρούμε στο δείκτη y ως περιεχόμενο τη διεύθυνση της μεταβλητής b και με την εντολή ***cout***<<*y; τυπώνεται η τιμή 3.5.

8 Αναφορές

Η αναφορά είναι ένα ψευδώνυμο μιας μεταβλητής. Συμβολίζεται με το χαρακτήρα &. Πρέπει κατά τη δήλωση της αναφοράς να γίνεται ταυτόχρονα και αρχικοποίηση προς τη μεταβλητή της οποίας αποτελεί ψευδώνυμο.Η δήλωση μιας αναφοράς έχει την ακόλουθη γενική μορφή:

τύπος &όνομα αναφοράς = μεταβλητή;

π.χ.

int& rx= x;

<u>Παράδειγμα 1-Πρόγραμμα με αναφορά προς μεταβλητή</u>

#***include*** <***iostream.h***>

void main()

{

int a=3,&p=a,b=a;//***δηλώνονται μεταβλητή a, εναλλακτικό όνομα p της a και μεταβλητή b με αρχική τιμή την a***

```
    p++;//τροποποίηση της p (αλλάζει την τιμή της a)
    cout<<a<<" "<<p<<endl; //τυπώνονται 4 (από την a) και 4 (από την p)
    b=b+10; // τροποποίηση της b (δεν αλλάζει την τιμή της a)
    cout<<a<<" "<<b<<endl; //τυπώνονται 4 (από την a) και 13 (από την b)
}
//Αποτελέσματα προγράμματος:
//4 4
//4 13
```

Παράδειγμα 2-Πρόγραμμα με αναφορά προς δείκτη

```
#include <iostream.h>

void main()
{
    int a=10,b=50;//ορίζονται οι μεταβλητές a και b
    int *q=&a;//ορίζεται ο pointer q προς τη μεταβλητή a
    int *&r=q;//ορίζεται αναφορά (εναλλακτικό όνομα) προς τον pointer q

    cout<<*r<<endl; //έμμεση τιμή με το εναλλακτικό όνομα είναι η τιμή της a
    r=&b//αλλάζουμε τη διεύθυνση που δείχνει ο r
    cout<<*q<<endl; //εκτυπώνεται σαν έμμεση τιμή του q η τιμή της b
}
//Αποτελέσματα προγράμματος:
//10 50
```

Παράδειγμα 3-Πρόγραμμα με πέρασμα τιμής, αναφοράς και δείκτη

```
#include <iostream.h>

void swap(int x,int y); //κλήση της swap με τιμή
void swap1(int *px,int *py); //κλήση της swap1 με δείκτες
void swap2(int &rx,int &ry); //κλήση της swap2 με αναφορές
```

```
void main()
{
    int x=5,y=10;

    cout<<"Before swap x = "<<x<<" and y ="<<y<<"\n";
    swap(x,y); //pass by value. Δημιουργούνται τοπικά αντίγραφα των x και y στη συνάρτηση
    cout<<"After swap x = "<<x<<" and y ="<<y<<"\n";

    x=5;
    y=10;
    cout<<"\n\nBefore swap1 x = "<<x<<" and y ="<<y<<"\n";
    swap1(&x,&y); //pass by reference with pointers. Οι διευθύνσεις των x και y περνάνε στο υποπρόγραμμα
    cout<<"After swap1 x = "<<x<<" and y ="<<y<<"\n";

    x=5;
    y=10;
    cout<<"\n\nBefore swap2 x = "<<x<<" and y ="<<y<<"\n";
    swap2(x,y); //pass by reference with references. Οι τιμές των x και y δεν καταχωρούνται σε τοπικές μεταβλητές αλλά σε ψευδώνυμα
    cout<<"After swap2 x = "<<x<<" and y ="<<y<<"\n";
}

void swap(int x,int y)
{
    int temp;

    temp=x;
    x=y;
    y=temp;
    cout<<"In swap x = "<<x<<" and y = "<<y<<"\n";
    }
```

```
        void swap1(int *px,int *py)
        {
        int temp;

        temp=*px;
        *px=*py;
        *py=temp;
        cout<<"In swap1 *px = "<<*px<<" and *py = "<<*py<<"\n";
}

void swap2(int &rx,int &ry)
{
        int temp;

        temp=rx;
        rx=ry;
        ry=temp;
        cout<<"In swap2 rx = "<<rx<<" and ry = "<<ry<<"\n\n";
}
//Αποτελέσματα προγράμματος
//Before swap x=5 and y=10
//In swap x=10 and y=5
//After swap x=5 and y=10

//Before swap1 x=5 and y=10
//In swap1 *px=10 and *py=5
//After swap1 x=10 and y=5

//Before swap2 x=5 and y=10
//In swap2 rx=10 and ry=5
//After swap2 x=10 and y=5
```

9 Δυναμική Διαχείριση Μνήμης

Η C++ επιτρέπει τη δυναμική διαχείριση της μνήμης. Αυτό σημαίνει ότι μπορούμε να δεσμεύουμε όση μνήμη (κύρια μνήμη) χρειαζόμαστε δυναμικά δηλαδή κατά τη διάρκεια εκτέλεσης του προγράμματος και όχι στατικά δηλαδή εκ των προτέρων. Η δυναμική δέσμευση της μνήμης γίνεται με την εντολή ***new***. Ως αποτέλεσμα η εντολή ***new*** επιστρέφει τη διεύθυνση του 1ου byte από τη μνήμη που δεσμεύεται. Για να αποδεσμεύσουμε τη μνήμη που δεν μας χρειάζεται χρησιμοποιούμε την εντολή ***delete***. Η δυναμική δέσμευση μνήμης είναι ιδιαίτερα χρήσιμη στους πίνακες γιατί δεσμεύοντας ακριβώς τη μνήμη που μας χρειάζεται για ένα πίνακα αποφεύγουμε 2 προβλήματα:

- Η δεσμευόμενη μνήμη να μην επαρκεί γα τον πίνακα
- Η δεσμευόμενη μνήμη να είναι υπερβολικά μεγάλη σε σχέση με αυτή που πραγματικά χρειάζεται

Παράδειγμα 1 -Πρόγραμμα με δυναμική κατανομή μνήμης για μονοδιάστατο πίνακα

```
#include <iostream.h>
#include <iomanip.h>//Βρίσκεται η setw()
#include <stdlib.h>//Βρίσκεται η exit()
void main()
{
int i,m,*a;//Η m δείχνει τη διάσταση του πίνακα

//ΒΑΣΙΚΟ!! Στους στατικούς πίνακες η διάσταση με την οποία δηλώνουμε τον πίνακα πρέπει να είναι σταθερά. Αντίθετα κατά τον ορισμό του δυναμικού πίνακα η διάσταση μπορεί να είναι μια μη σταθερή μεταβλητή, αλλά να διαβάζεται κατά την εκτέλεση του προγράμματος όπως γίνεται για το m

        cout<<"Δώσε το μέγεθος του πίνακα: ";
        cin>>m;//Το μέγεθος του πίνακα διαβάζεται κατά τη διάρκεια εκτέλεσης του προγράμματος δεν δίνεται εκ των προτέρων

        a=new int[m];//Δεσμεύεται χώρος για τον πίνακα στο heap
        //Βασικό! Η ισοδύναμη εντολή στη C είναι a=(int)malloc(m*sizeof(int));
        if (!a)//Εάν δεν έγινε κατανομή χώρου
        {
                cout<<"ΛΑΘΟΣ ΚΑΤΑΝΟΜΗΣ";
                exit(1);
        }
        for (i=0;i<m;i++)
        {
                cout<<"Δώσε το "<<i+1<<" στοιχείο του πίνακα: ";
                cin>>a[i];
        }
        for (i=0;i<m;i++)
                cout<<setw(4)<<*(a+i)<<" ";//Εκτυπώνονται τα στοιχεία του πίνακα
                cout<<'\n';
        delete []a;//Απελευθερώνεται η μνήμη του heap
        //Βασικό! Η ισοδύναμη εντολή στη C είναι free(a);
}
```

Παράδειγμα 2 - Πρόγραμμα με δυναμική κατανομή μνήμης για δισδιάστατο πίνακα

```
#include <iostream.h>
#include <iomanip.h>//για την setw()
#include <stdlib.h>//για την exit()
void main()
{
int i,j,m,n;

//ΠΡΟΣΟΧΗ! Όπως στο μονοδιάστατο έτσι και στο δι-διάστατο πίνακα το πλήθος των γραμμών και των στηλών του δεν ορίζεται εκ των προτέρων (στατικά) αλλά διαβάζονται τη στιγμή εκτέλεσης του προγράμματος (δυναμικά)

        cout<<"Δώσε το πλήθος γραμμών του πίνακα: ";
        cin>>m;

        cout<<"Δώσε το πλήθος στηλών του πίνακα:  ";
        cin>>n;

        //Ο δι-διάστατος πίνακας δεν μπορεί να αναπαρασταθεί απευθείας στο heap οπότε για την αναπαράσταση του γίνονται τα ακόλουθα βήματα:

//Βήμα 1
        int *p=new int[m*n];//Ορίζεται ο pointer p προς ακεραίους, κρατούνται θέσεις μνήμης στο heap για m*n ακεραίους και η διεύθυνση του πρώτου ακέραιου τοποθετείται στο p

//Βήμα 2
        int **a=new int *[m];//Ορίζεται ο a σαν δείκτης προς δείκτη σε ακέραιο, κρατούνται m θέσεις στο heap για διευθύνσεις γραμμών (όσες είναι και οι γραμμές) και η πρώτη διεύθυνση αυτών των θέσεων τοποθετείται στο a

        if (!p ||!a)//Εάν δεν έγινε κάποια κατανομή μνήμης έξοδος
                exit(1);

//Βήμα 3
        for(i=0;i<m;i++)
                a[i]=p+i*n;//Προσδιορίζουμε ότι η διεύθυνση a[0] (πρώτο στοιχείο του a) θα έχει τιμή p (διεύθυνση πρώτης γραμμής του πίνακα a), η διεύθυνση a[1] (δεύτερο στοιχείο του πίνακα a) θα είναι η διεύθυνση μετά από n θέσεις από την p στον πίνακα p (δηλαδή η διεύθυνση στον p όπου βρίσκεται το πρώτο στοιχείο της επόμενης γραμμής) κ.λ.π.

        cout<<"Διάβασμα του πίνακα"<<endl;
        for (i=0;i<m;i++)
                for (j=0;j<n;j++)
                {
                        cout<<"Διάβασε το "<<i+1<<" "<<j+1<<" στοιχείο του πίνακα: ";
                        cin>>a[i][j];
                }

        cout<<"Εκτύπωση του πίνακα"<<endl;
```

```
        for (i=0;i<m;i++)
        {
                for (j=0;j<n;j++)
                        cout<<setw(4)<<a[i][j]<<" ";
                cout<<'\n';
        }

        delete []p;//Απελευθερώνεται η μνήμη όπου είναι οι ακέραιοι
        delete []a;//Απελευθερώνεται η μνήμη όπου είναι οι διευθύνσεις των πρώτων στοιχείων των γραμμών
}
```

10 Σύνθετοι Τύπου Δεδομένων

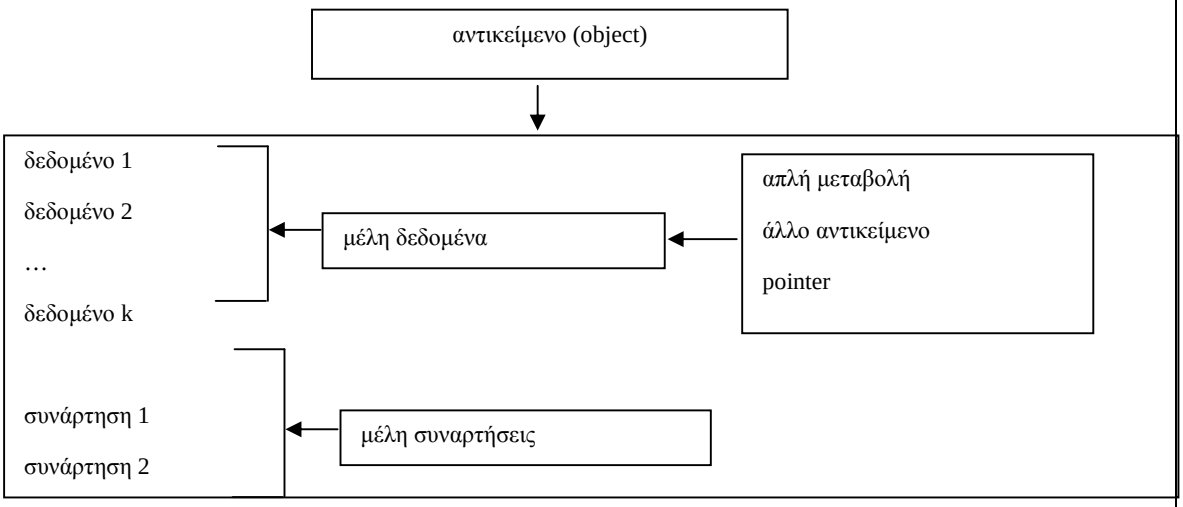
10.1 Βασικές Έννοιες

10.1.1 Τα Αντικείμενα (objects) και τα μέλη τους

Ένα από τα βασικά χαρακτηριστικά της C++ είναι η δυνατότητα δημιουργίας **αντικειμένων** (objects). Τα αντικείμενα είναι χώρος μνήμης όπου κρατούνται ενότητες από δεδομένα διαφόρων τύπων και συναρτήσεις σχετικές με τα δεδομένα.

αντικείμενα = χώρος μνήμης με δεδομένα + συναρτήσεις για τα δεδομένα

Τα δεδομένα ενός αντικειμένου μπορεί να είναι μεταβλητές απλών τύπων, πίνακες (arrays), άλλα αντικείμενα ή μεταβλητές διευθύνσεων (pointers) προς μεταβλητές, συναρτήσεις ή άλλα αντικείμενα. **Οι συναρτήσεις και τα δεδομένα ενός αντικειμένου ονομάζονται μέλη** (members) του αντικειμένου. Θέτοντας τα δεδομένα και τις συναρτήσεις που αφορούν αυτά σε αντικείμενα (objects) επιτυγχάνουμε να προφυλάσσουμε τα δεδομένα από ανεπιθύμητες τροποποιήσεις και να αναπτύσσουμε μεγάλα προβλήματα με ευκολία.



Τα αντικείμενα σε ένα πρόγραμμα μπορεί να είναι **επώνυμα** (να έχουν όνομα) ή ανώνυμα (να προσδιορίζονται από δείκτη.

10.1.2 Κατηγορίες και Τύποι Αντικειμένων

Η C++ προσφέρει τη δυνατότητα δημιουργίας τριών κατηγοριών αντικειμένων. Αντικείμενα κατηγορίας **ένωσης** (union), κατηγορίας **δομής** (structure) και κατηγορίας **κλάσης** (***class***). Για κάθε μια κατηγορία μπορούμε να δημιουργήσουμε αντικείμενα διαφορετικού τύπου. Ο τύπος προσδιορίζει το "καλούπι" πάνω στο οποίο δημιουργούνται τα αντικείμενα και συνήθως χαρακτηρίζεται στο πρόγραμμα από ένα όνομα. Για κάθε τύπο (μιας κατηγορίας) μπορούμε να δημιουργούμε ένα ή περισσότερα αντικείμενα.

10.1.3 Βήματα για Δημιουργία Αντικειμένων

Για να χρησιμοποιήσουμε αντικείμενα στο πρόγραμμά μας ακολουθούμε 2 φάσεις:

- Ορίζουμε ένα σύνθετο τύπο ο οποίος θα ανήκει σε μια από τις κατηγορίες ένωσης (union), δομή (structure), κλάση (***class***).
- Ορίζουμε αντικείμενα (objects) με βάση τον τύπο.

Για παράδειγμα μπορεί να χρησιμοποιούμε στο πρόγραμμά μας δύο αντικείμενα διαφορετικού τύπου της κατηγορίας κλάσης στα οποία το ένα να έχει μέλη ένα ακέραιο και δύο συναρτήσεις. και το άλλο να έχει δύο μέλη ακεραίους και μια συνάρτηση. Τα αντικείμενα έχουν οπωσδήποτε κάποιο τύπο (επώνυμο ή ανώνυμο).

10.1.4 Βασικές Διαφορές κατηγοριών Αντικειμένων

Η διαφορά των κατηγοριών κλάσεων και δομών στη C++ δεν είναι ουσιαστική. Αφορά μόνο τον καθιερωμένο τρόπο προσπέλασης των μελών των αντικειμένων.

10.1.5 Προσπέλαση και Άδειες Προσπέλασης Μελών

Σε ένα πρόγραμμα που έχουμε ορίσει ένα αντικείμενο με κάποια μέλη μπορούμε να προσπελαύνουμε τα μέλη, μέσω του ονόματός του αντικειμένου ή μέσω ενός pointer στον οποίο εκχωρούμε την διεύθυνση του αντικειμένου.
Ας υποθέσουμε ότι έχουμε ορίσει ένα αντικείμενο με όνομα a και με μέλη μια συνάρτηση με όνομα f και μια μεταβλητή με όνομα k. Ας υποθέσουμε ακόμη ότι, έχουμε ορίσει ένα pointer p (προς αντικείμενα του τύπου του a) και έχουμε εκχωρήσει (με p=&a) στον p την διεύθυνση του αντικειμένου a. Τότε η προσπέλαση των μελών του a γίνεται:

α) ***χρησιμοποιώντας το όνομα και τον τελεστή "." (τελεία) με τις παραστάσεις***:

- a.k (για τον προσδιορισμό του μέλους δεδομένου)
- a.f() (για την κλήση της συνάρτησης μέλους)

β)**χρησιμοποιώντας τον pointer και τον τελεστή "->"με τις παραστάσεις**:

- p->k (για τον προσδιορισμό του μέλους δεδομένο)
- p->f () (για την κλήση της συνάρτησης μέλους)

Στη C++ τα μέλη των αντικειμένων μπορεί να χαρακτηρίζονται (από τον τύπο των αντικειμένων) σαν **δημόσια** (***publc***) ή **ιδιωτικά** (***private***) ή **προστατευμένα** (***protected***).

Σημείωση:
Οι τελεστές τελεία "." και "->" χρησιμοποιούνται μόνο για την προσπέλαση μελών που είναι δημόσια (***publc***). Δεν επιτρέπεται η χρησιμοποίησή τους σε μέλη ιδιωτικά (***private***) ή προστατευμένα (***protected***).

10.1.6 Ορισμός Συνθετών Τύπων Αντικειμένων

Για να ορίσουμε ένα αντικείμενο πρέπει προηγούμενα να ορίσουμε τον τύπο (το καλούπι) του αντικειμένου τον οποίο αποκαλούμε **σύνθετο τύπο δεδομένων**. Στη C++ έχουμε την δυνατότητα να ορίσουμε τρεις κατηγορίες σύνθετων τύπων. Σύνθετους τύπους (καλούπια) για **ενώσεις**, για **δομές** και για **κλάσεις**. Οι σύνθετοι τύποι (για κάθε κατηγορία) ορίζονται με τον ίδιο τρόπο και με δηλώσεις στις οποίες διακρίνουμε τρία τμήματα: **Επικεφαλίδα, Σώμα τύπου, Λίστα αντικειμένων**

- Στην επικεφαλίδα αναφέρεται η κατηγορία του τύπου με μια από τις δεσμευμένες λέξεις union (για ενώσεις), ***struct*** (για δομές), ***class*** (για κλάσεις) και το όνομα του τύπου. (Το όνομα του τύπου μπορεί να μην υπάρχει στην περίπτωση που τα αντικείμενα που θα δημιουργηθούν, είναι μόνο εκείνα που αναφέρονται στην λίστα αντικειμένων).

- Στο σώμα τύπου διακρίνουμε μέσα σε μπλοκ { } δηλώσεις με τις οποίες:
  1. περιγράφονται τα μέλη δεδομένα
  2. δηλώνονται τα πρωτότυπα ή ορίζονται οι συναρτήσεις μέλη.
  3. προσδιορίζονται οι άδειες προσπέλασης των μελών (μια από τις δεσμευμένες λέξεις: ***publc*****:** ή ***private*****:** ή ***protected*****:** πριν από μέλος ή τα μέλη)

- Στη λίστα αντικειμένων αναφέρονται ονόματα αντικειμένων. Τα ονόματα αντικειμένων αν είναι πολλά διαχωρίζονται με κόμμα. Η λίστα επιτρέπεται να είναι κενή. Αν η λίστα είναι κενή, ο σύνθετος τύπος πρέπει να έχει οπωσδήποτε όνομα και τα αντικείμενα να ορίζονται σε άλλα σημεία του προγράμματος, με εντολές δήλωσης τύπου: όνομα_σύνθετου τύπου όνομα_αντικειμένου;

Οι συναρτήσεις μέλη του τύπου μπορεί να ορίζονται μέσα στο μπλοκ σώματος του τύπου ή μέσα στο μπλοκ να αναφέρεται μόνο το πρωτότυπό τους και ο ορισμός να δίνεται εκτός του μπλοκ. Εάν μια συνάρτηση ορίζεται μέσα στο μπλοκ (του σύνθετου τύπου) χαρακτηρίζεται σαν εντός γραμμής (**inline**). Μια συνάρτηση **inline** αναπτύσσει κώδικα (εντολές) σε κάθε σημείο του προγράμματος που θα κληθεί. **Χρησιμοποιούνται inline συναρτήσεις όταν ο κώδικας είναι μικρός για να επιταχύνεται η εκτέλεση του προγράμματος** (διότι δεν γίνεται αναζήτηση του σημείου εισόδου της συνάρτησης και καθυστέρηση για μεταφορά ελέγχου του προγράμματος). Όταν ο κώδικας είναι "μεγάλος" τότε με inline συνάρτηση θα έχουμε "μεγάλη" αύξηση του πλήθος εντολών του προγράμματος άρα και η πιθανή επιβράδυνση της εκτέλεσης. Δεν επιτρέπεται μια inline συνάρτηση να έχει στον ορισμό της ανακυκλώσεις (***for***, ***while***, ***do while***) ή διακλαδώσεις (***if***, ***case***). Μια τέτοια συνάρτηση θα ορισθεί εκτός γραμμής (out of line). Στο μπλοκ ορισμού του σύνθετου τύπου θα δοθεί μόνο το πρωτότυπο της συνάρτησης και ο ορισμός θα γίνει εκτός μπλοκ. Μια συνάρτηση μέλος ορίζεται εκτός του μπλοκ περιγραφής των μελών με εντολές της μορφής:
τύπος_συνάρτηση όνομα_σύνθετου_τύπου :: όνομα_συνάρτησης {…εντολές…}

Σημείωση

Μια συνάρτηση μέλος μπορεί να οριστεί σαν inline και εκτός του μπλοκ ορισμού του τύπου, αν βάλουμε πριν τον ορισμό της τη λέξη κλειδί inline με εντολές της μορφής:
inline τύπος_συνάρτησης όνομα_τύπου :: όνομα_συνάρτησης {…εντολές…}

10.1.7 Η Ενθυλάκωση (Encapsulation) Μελών

Δημιουργώντας ένα σύνθετο τύπο έχουμε τη δυνατότητα να προσδιορίζουμε κάποια μέλη σαν **ιδιωτικά** (***private***). Αυτό παρέχει την ευκολία να αποκρύπτουμε τις λεπτομέρειες (βοηθητικές ή μεταβλητές συναρτήσεις) που συμμετέχουν στην εκτέλεσης της διαδικασίας που μας ενδιαφέρει. Έχουμε όπως λέμε τη δυνατότητα πιο καθαρής διασύνδεσης (interface) με το αντικείμενο και επομένως αποφυγή λαθών λόγω πιθανώς μη επιτρεπτών παρεμβάσεων στα μέλη του.

Με το χαρακτηρισμό ενός μέλους **σαν ιδιωτικό επιτυγχάνουμε την απόκρυψη του μέλους (data hiding) από αυτόν που χρησιμοποιεί το αντικείμενο**. **Είναι σαν να τοποθετούμε τα ιδιωτικά μέλη σε μια ειδική θέση και να τα προφυλάσσουμε από εξωτερικές επιδράσεις**. Η ιδιότητα αυτή αναφέρεται σαν **ενθυλάκωση** (encapsulation) των μελών στο αντικείμενο.

10.1.8 Καθιερωμένες Προσπελάσεις

Έχουμε αναφέρει, ότι οι άδειες προσπέλασης που μπορούμε να εκχωρήσουμε σε μέλη αντικειμένων κατά τον προσδιορισμό ενός σύνθετου τύπου, είναι τρεις:

publc (δημόσια), ***private*** (ιδιωτικά) και ***protected*** (προστατευμένα)

@ Όταν το μέλος είναι *publc* (δημόσιο) επιτρέπεται η προσπέλασή του με έκφραση της μορφής a.k

@ Όταν το μέλος είναι *private* (ιδιωτικό) ή *protected* (προστατευμένο) μια τέτοια προσπέλαση δεν είναι επιτρεπτή

Ο χαρακτηρισμός ενός μέλους σαν ***protected*** χρησιμοποιείται σε παραγόμενους σύνθετους τύπους. Η C++ έχει καθιερωμένες (default) άδειες προσπέλασης των μελών, για αντικείμενα των τύπων των κατηγοριών ενώσεων, δομών και κλάσεων σε περίπτωση που δεν χαρακτηρίζουμε άμεσα την άδεια προσπέλασης. Εάν δεν καθορίζουμε άμεσα την άδεια προσπέλασης μέλους (με τις δεσμευμένες λέξεις ***publc*** ή ***protected*** ή ***private***) τότε κατά τα καθιερωμένα:

· τα μέλη αντικειμένων των κατηγοριών ενώσεων (union) και δομών (structure) είναι ***publc*** (δημόσια)

· τα μέλη αντικειμένων της κατηγορίας κλάση (***class***) είναι ***private*** (ιδιωτικά)

Βασική Διαφορά

Η μοναδική διαφορά μεταξύ κλάσης (***class***) και δομής (structure) είναι ότι τα μέλη αντικειμένων της πρώτης κλάσης προκαθορίζονται ***private*** (ιδιωτικά) ενώ της δομής ***publc*** (δημόσια).

10.1.9 Παραδείγματα Ενώσεων, Δομών και Κλάσεων

Σε κάθε στήλη του πίνακα που ακολουθεί ορίζουμε και ένα σύνθετο τύπο δεδομένων. Στην $1^{η}$ στήλη ορίζεται ανώνυμος τύπος κατηγορίας ένωση και αντικείμενα a, b. Στη $2^{η}$ στήλη ορίζεται τύπος s κατηγορίας δομή. Στην $3^{η}$ στήλη τύπος c κατηγορίας κλάση με αντικείμενο το r.

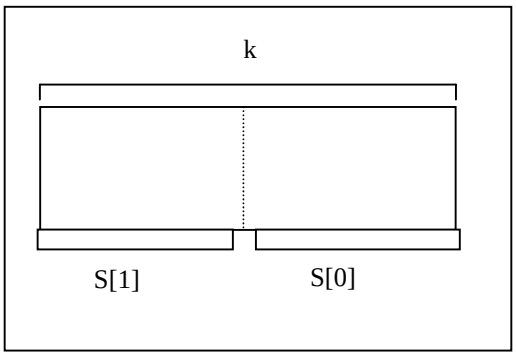
union	***struct*** s	***class*** c
{	{	{
int k;	***int*** k;	***int*** k;
char s [2];	***void*** show () {***cout*** <<k;}	***publc***:
} a, b;	};	***void*** put (***int*** x) {x=x;}
		void show ()
		} r;
		void c:: show () {***cout*** <<k;}

Ειδικότερα για κάθε στήλη διαπιστώνουμε:

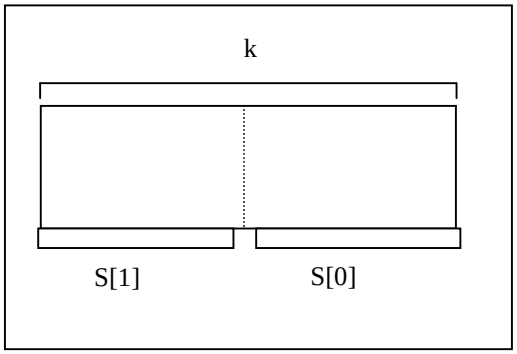
Στην 1η στήλη

Τα αντικείμενα a, b του τύπου έχουν δύο μέλη δεδομένα. Το μέλος k (ακέραιος) και το μέλος s (πίνακας με 2 χαρακτήρες). Επειδή ο τύπος είναι ανώνυμος δεν είναι δυνατόν να οριστούν άλλα αντικείμενα εκτός των a, b. **Τα μέλη των αντικειμένων αυτού του τύπου είναι *publc* (κατά τα καθιερωμένα για τις ενώσεις).** Τα μέλη του a μοιράζονται τον ίδιο χώρο μνήμης και το αυτό συμβαίνει για τα μέλη του αντικειμένου b.

αντικείμενο a



αντικείμενο b



Το αντικείμενο a έχει τα μέλη a.k και τον πίνακα με στοιχεία a.s [0] και a.s [1] που μοιράζονται τον ίδιο χώρο μνήμης. Το b έχει επίσης τα μέλη b.k και πίνακα με στοιχεία b.s[0] και b.s[1]. Μπορούμε μέσα από μια συνάρτηση να τοποθετούμε στο ένα μέλος του αντικειμένου a (ή του b) μια τιμή και να εμφανίζουμε την τιμή του άλλου μέλους π.χ. να δίνουμε τιμή στο k και να εκτυπώνουμε τα bytes από τα στοιχεία του s ή το αντίστροφο να δίνουμε τους χαρακτήρες στο s και να παίρνουμε την τιμή του διαμορφούμενου ακεραίου στο k (δηλαδή να δώσουμε τις εντολές a.k =100; ***cout*** <<a.s [0] << " "<<a.s [1] ή τις εντολές b.s[0] ='A' ;b.s [1] ='B'; ***cout*** << b.k;)

Στη 2η στήλη

Με τον ορισμό του τύπου s δεν ορίζονται και αντικείμενα (η λίστα είναι κενή). Αν ορισθούν κάπου αλλού στο πρόγραμμά μας αντικείμενα του τύπου s, θα έχουν μέλη ένα ακέραιο k και μια inline συνάρτηση show() που θα εμφανίζει την τιμή του ακεραίου. **Και τα δύο μέλη είναι *publc* (δημόσια) κατά τα καθιερωμένα για τις δομές**. Για να ορίσουμε αντικείμενα x,y του τύπου s θα δώσουμε εντολή της μορφής: s x,y; Μετά τον ορισμό των χ, y επιτρέπονται οι εντολές:
x.k =100; y.k=200; x.show (); y.show ();
Οι δύο πρώτες δίνουν τιμή στο μέλος k κάθε αντικειμένου και οι δύο επόμενες εμφανίζουν την τιμή του k για κάθε αντικείμενο.

Στην 3η στήλη

Μαζί με τον τύπο c ορίζεται και το αντικείμενο r. Το αντικείμενο r έχει τρία μέλη. Ένα ακέραιο k μέλος **ιδιωτικό** (***private* –κατά τα καθιερωμένα στις κλάσεις**) και δύο **δημόσια** μέλη συναρτήσεις put () και show () (**επειδή υπάρχει η δήλωση *publc*:**). Η συνάρτηση put() είναι inline ενώ η συνάρτηση show () όχι. Λόγω των αδειών προσπέλασης των μελών δεν

επιτρέπεται να χρησιμοποιήσουμε προσπέλαση των αδειών προσπέλασης της μορφής r.k, αλλά μπορούμε να προσπελάσουμε τα μέλη συναρτήσεις. Το μέλος συνάρτησης put δέχεται παράμετρο ένα ακέραιο και τον τοποθετεί σαν τιμή του μέλους k. Το μέλος συνάρτηση show εμφανίζει την τιμή του μέλους k. Έτσι μπορούμε να χρησιμοποιήσουμε τις εντολές:
r.put (100); και r.show (); //η πρώτη θέτει στο k το 100 η δεύτερη το εμφανίζει.

11 Κατασκευαστές και Καταστροφέας

11.1 Συναρτήσεις Κατασκευαστές (constructors)

11.1.1 Ο Καθιερωμένος Κατασκευαστής (default constructor)

Γνωρίζουμε ότι ορίζοντας σε ένα πρόγραμμα μια κλάσης (γενικότερα δομή ή ένωση) μπορούμε στη συνέχεια να δημιουργήσουμε αντικείμενα αυτού του τύπου. Η δημιουργία των αντικειμένων πραγματοποιείται μέσω μιας συνάρτησης που προσφέρεται αυτόματα από την C++ με τον ορισμό της κλάσης και ονομάζεται καθιερωμένος κατασκευαστής (default constructor). Η καθιερωμένη συνάρτηση κατασκευής (default constructor), δημιουργεί ένα αντικείμενο (object), αφιερώνει χώρο μνήμης για το αντικείμενο αλλά δεν τοποθετεί συγκεκριμένες τιμές στα μέλη-δεδομένα του αντικειμένου. Έτσι αν μετά την δημιουργία του αντικειμένου ζητήσουμε εμφάνιση των τιμών των μελών δεδομένων του αντικειμένου θα πάρουμε απροσδιόριστες τιμές. Για να τοποθετήσουμε συγκεκριμένες τιμές στα δεδομένα μέλη ενός αντικειμένου μετά την δημιουργία του μπορούμε να χρησιμοποιήσουμε:

- Εντολές εκχώρησης (assignment) τιμών εάν το δεδομένο μέλος έχει δυνατότητα δημόσιας προσπέλασης (είναι ***publc***) ή
- Δημόσιες συναρτήσεις μέλη που έχουν οριστεί στην κλάση για τον σκοπό αυτό.

11.1.2 Δημιουργία Κατασκευαστή (constructor)

Η C++ μας παρέχει τη δυνατότητα να δημιουργούμε (ορίζοντας μια κλάση) δικές μας συναρτήσεις κατασκευαστές (constructors) οι οποίες αντικαθιστούν την καθιερωμένη συνάρτηση κατασκευαστή (default constructor) κάθε φορά που δημιουργούνται αντικείμενα στο πρόγραμμά μας. Η δημιουργία μιας συνάρτησης κατασκευαστή (constructor) γίνεται με ορισμό συνάρτησης μέλους μέσω εντολής, που έχει την μορφή:

όνομα_κλάσης (λίστα παραμέτρων) {…εντολές…;)

Στον ορισμό της συνάρτησης κατασκευαστή δεν χρησιμοποιούμε τύπο συνάρτησης και η συνάρτηση πρέπει να είναι ***publc***. Η λίστα παραμέτρων καθορίζει τις αναγκαίες πληροφορίες που θα επισυνάπτουμε στα ονόματα των αντικειμένων (που δημιουργούμε με τον constructor), και οι εντολές αναφέρονται στο τι θα εκτελείται κατά την δημιουργία του αντικειμένου (συνήθως εκχωρήσεις τιμών των παραμέτρων στα μέλη του αντικειμένου).

11.1.3 Λόγοι που επιβάλλουν δημιουργία κατασκευαστή

Ο βασικός λόγος που επιβάλλει την δημιουργία ενός κατασκευαστή (constructor) είναι η εκτέλεση κάποιων εντολών (εκτός από τις καθιερωμένες), κατά τον χρόνο που δημιουργούνται τα αντικείμενα. Για παράδειγμα όταν κατά την δημιουργία ενός αντικειμένου θέλουμε:

α)να εκτυπώνονται μηνύματα ή
β)να δίνονται αρχικές τιμές στις μεταβλητές, που είναι μέλη του αντικειμένου (ή και να τροποποιούνται τιμές μεταβλητών μη μελών) ή
γ)να εκτελούνται συναρτήσεις (μέλη του αντικειμένου ή και μη μέλη)

τότε πρέπει να ορίσουμε δικό μας κατασκευαστή (constructor).

11.1.4 Πότε καλείται ένας κατασκευαστής (constructor)

Ο κατασκευαστής (constructor) που έχουμε ορίσει σε μια κλάση, χρησιμοποιείται σε 2 περιπτώσεις:

- Για να ορίσουμε ένα νέο αντικείμενο και
- Για να επαναπροσδιορίσουμε ένα ήδη ορισμένο αντικείμενο

Ο ορισμός ενός νέου αντικειμένου γίνεται με τους εξής τρόπους:
α)όνομα_κλάσης όνομα_αντικειμένου (λίστα_τιμών_παραμέτρων); ή
β)όνομα_κλάσης όνομα_αντικειμένου = όνομα_κλάσης (λίστα_τιμών_παραμέτρων);

Ακόμη σε περίπτωση που η συνάρτηση κατασκευαστής έχει μία μόνο παράμετρο το νέο αντικείμενο μπορεί να οριστεί και ως εξής:
γ)όνομα_κλάσης όνομα_αντικειμένου = τιμή_παραμέτρου;

Σημείωση

Η περίπτωση (γ) είναι μια ειδική περίπτωση και εφαρμόζεται μόνο όταν ο constructor έχει μία μόνο παράμετρο. Αν ο constructor ορίζεται με περισσότερες από μία παραμέτρους δεν εφαρμόζεται.

Ο επαναπροσδιορισμός ενός αντικειμένου γίνεται με τους επόμενους τρόπους:
α)όνομα_αντικειμένου =όνομα_κλάσης (λίστα_τιμών_παραμέτρων) και στην περίπτωση που έχουμε constructor με μια μόνο παράμετρο μπορούμε να επαναπροσδιορίζουμε ένα ήδη ορισμένο αντικείμενο με εντολή της μορφής:
β) όνομα_αντικειμένου = τιμή_παραμέτρου;

11.1.5 Υπερφορτώσεις σε Κατασκευαστές

Στη δημιουργία κλάσεων έχουμε την δυνατότητα να υπερφορτώνουμε (overloading) τις συναρτήσεις κατασκευαστές (constructors) δηλαδή να δημιουργούμε πολλές συναρτήσεις με όνομα το όνομα της κλάσης. Η δυνατότητα αυτή υπόκειται στους γενικούς περιορισμούς υπερφόρτωσης συναρτήσεων δηλαδή πρέπει να έχουμε διαφορετικό πλήθος παραμέτρων ή διαφορές στον τύπο ή διαφορές στην σειρά ως προς την οποία εμφανίζονται οι διάφοροι τύποι των παραμέτρων. Υπερφόρτωση constructor ονομάζεται η δήλωση του ίδιου constructor με διαφορετικά ορίσματα. Αυτό γίνεται όταν θέλουμε ο ίδιος constructor να επιτελεί διαφορετικές λειτουργίες ανάλογα με τις τιμές που λαμβάνει όταν δηλώνουμε αντικείμενα μιας συγκεκριμένης κλάσης. Η δυνατότητα αυτή υπόκειται στους γενικούς περιορισμούς υπερφόρτωσης συναρτήσεων δηλαδή πρέπει να έχουμε διαφορετικό πλήθος παραμέτρων ή διαφορές στον τύπο ή διαφορές στη σειρά με την οποία εμφανίζονται οι διάφοροι τύποι των παραμέτρων.

11.1.6 Καθιερωμένες Παράμετροι σε Κατασκευαστές

Ορίζοντας ένα constructor μπορούμε να δίνουμε στις παραμέτρους του και καθιερωμένες (default) τιμές. Έχοντας ορίσει default παραμέτρους μπορούμε να δημιουργούμε αντικείμενα χρησιμοποιώντας την ίδια συνάρτηση κατασκευαστή με διαφορετικό πλήθος παραμέτρων. Φυσικά και εδώ, όπως και γενικά στις απλές συναρτήσεις, οι καθιερωμένες παράμετροι πρέπει να τοποθετούνται μετά τις κανονικές παραμέτρους (αυτές που δεν παίρνουν default τιμές). Ορίζοντας ένα constructor μπορούμε να έχουμε στις παραμέτρους του και

καθιερωμένες (default) παραμέτρους. Έχοντας ορίσει default παραμέτρους μπορούμε να δημιουργούμε αντικείμενα χρησιμοποιώντας την ίδια συνάρτηση κατασκευαστή με διαφορετικό πλήθος παραμέτρων. Φυσικά και εδώ όπως και στις απλές συναρτήσεις οι καθιερωμένες παράμετροι πρέπει να τοποθετούνται μετά τις κανονικές παραμέτρους (αυτές που δεν παίρνουν default τιμές). Επίσης να τονίσουμε ότι είναι εντελώς διαφορετική περίπτωση να έχουμε default παραμέτρους σε constructor από την υπερφόρτωση του constructor, διότι εδώ έχουμε μόνο μια συνάρτηση constructor, ενώ στην υπερφόρτωση του constructor έχουμε πολλές συναρτήσεις constructor.

11.1.7 Αρχικές Τιμές σε Πίνακα Αντικειμένων

Για να δώσουμε αρχικές τιμές σε ένα πίνακα που τα στοιχεία του είναι αντικείμενα μιας κλάσης θα πρέπει να έχουμε ορίσει στην κλάση συνάρτηση κατασκευαστή (constructor). Σημειώνουμε ότι μια συνάρτηση κατασκευαστή με μια παράμετρο χρησιμοποιούμε για δημιουργία αντικειμένων με αρχικές τιμές κατά διαφορετικό τρόπο από εκείνη με δύο ή περισσότερες παραμέτρους. Για παράδειγμα αν έχουμε κλάση data και σε αυτήν έχουμε ορίσει constructor με πρωτότυπο data(***int*** k); τότε μπορούμε να ορίσουμε αντικείμενο a με αρχική τιμή το 10 ως εξής: data a=10; Σε περίπτωση που η συνάρτηση κατασκευαστής έχει δύο ή περισσότερες παραμέτρους π.χ. έχει πρωτότυπο data (***int*** k, ***char*** *s); τότε για να ορίσουμε αντικείμενο a με αρχικές τιμές θα πρέπει να δώσουμε δήλωση της μορφής data a=data (10, "age");

11.1.8 Συνάρτηση Καταστροφέας (Destructor)

Τα αντικείμενα που δημιουργούνται σε ένα πρόγραμμα καταστρέφονται αυτόματα όταν το πρόγραμμα εκτελέσει όλες τις εντολές του μπλοκ μέσα στο οποίο δημιουργήθησαν τα αντικείμενα (γενικότερα όταν παύσουν τα αντικείμενα να είναι εν ζωή).
Η καταστροφή των αντικείμενων συνεπάγεται την απελευθέρωση του χώρου μνήμης που κατείχαν τα αντικείμενα και δυνατότητα χρησιμοποίησης του χώρου αυτού για υπόλοιπες απαιτήσεις του προγράμματος. Εάν δημιουργούμε στο πρόγραμμά μας αντικείμενα σε δυναμική μνήμη (στο heap) την ευθύνη της καταστροφής των αντικειμένων (δηλαδή την απελευθέρωση του χώρου μνήμης) την έχουμε εμείς και η καταστροφή δεν γίνεται αυτόματα. Φυσικά και στην περίπτωση αυτή ο χώρος μνήμης απελευθερώνεται μετά το πέρας του προγράμματος, αλλά ο χώρος μνήμης μερικές φορές μπορεί να μη επαρκεί κατά την εκτέλεση του προγράμματος επειδή κρατείται για αντικείμενα που πλέον δεν χρειάζονται.
Η C++ μας προσφέρει τη δυνατότητα να δημιουργούμε συνάρτηση καταστροφέα αντικειμένου (destructor) που απελευθερώνει τον χώρο μνήμης του αντικειμένου μόλις το αντικείμενο παύσει να είναι εν ζωή. Η συνάρτηση καταστροφέας (destructor) που δημιουργούμε εκτελεί το έργο της (απελευθέρωση μνήμης) σε κάθε περίπτωση που το αντικείμενο πρέπει να καταστραφεί είτε αυτό δημιουργείτε σε ελεύθερη μνήμη είτε όχι.

11.1.9 Δημιουργία Καταστροφέα (Destructor)

Η συνάρτηση καταστροφέας είναι μια ειδικού τύπου συνάρτηση (όπως ο constructor) δεν έχει τύπο, δεν παίρνει παραμέτρους, το όνομά της είναι το ίδιο με την κλάση (ή ένωση ή δομή) στην οποία ορίζεται και έχει πριν το όνομα του χαρακτήρα '~'. Δηλαδή ένας destructor ορίζεται με εντολές της μορφής: **~ όνομα_κλάσης () {…εντολές…}**
Οι εντολές που τοποθετούμαι στο μπλοκ ορισμού του destructor είναι συνήθως εντολές εκτύπωσης μηνυμάτων ή εντολές απελευθέρωσης δυναμικής μνήμης.

Σημείωση:

Δεν χρειάζεται να δημιουργήσουμε δικό μας destructor σε περιπτώσεις που δεν κατανέμουμε δυναμική μνήμη σε αντικείμενα. Σε περιπτώσεις που κατανέμουμε δυναμική μνήμη σε αντικείμενα ο καταστροφέας (destructor) είναι απαραίτητος.

Παράδειγμα 1 Εξορισμού (Καθιερωμένες) παράμετροι σε κατασκευαστή

```
#include<iostream.h>
#include <math.h>

class shapes
{
publc:
        shapes(int x1=0,int x2=0,int y1=0,int y2=0)
        {
                if (x1!=0 && x2!=0 && y1!=0 && y2!=0)
                {
                        dist=sqrt(pow((x2-x1),2)+pow((y2-y1),2));
                        cout<<"Distance "<<dist<<endl;
                }

                if (x1!=0 && x2!=0 && y1!=0 && y2==0)
                {
                        area=(x1+x2)*y1/2;
                        cout<<"Area of trapezoid = "<<area<<endl;
                }

                if (x1!=0 && x2!=0 && y1==0 && y2==0)
                {
                        area=x1*x2;
                        cout<<"Area of rectangle = "<<area<<endl;
                }

                if (x1!=0 && x2==0 && y1==0 && y2==0)
                {
                        area=x1*x1;
                        cout<<"Area of square = "<<area<<endl;
                }

                if (x1==0 && x2==0 && y1==0 && y2==0)
                {
                        int i,j;

                        for(i=1;i<=5;i++)
                        {
                                for(j=1;j<=5-i;j++)
                                        cout<<"*";
                                cout<<endl;
                        }
                }
        }
```

```
private:
        double dist;
        int area;
};

void main()
{
        int x1,y1,x2,y2,z,ch;

        cout<<"1.Trapezoid."<<endl;
        cout<<"2.Rectangle."<<endl;
        cout<<"3.Square."<<endl;
        cout<<"4.Distans."<<endl;
        cout<<"5.Draw shape."<<endl;

        cout<<"Give your choice: ";
        cin>>ch;

        if (ch==1)
        {
                cout<<"Give base1 and base2 and height of trapezoid: ";
                cin>>x1>>y1>>z;
                shapes s(x1,y1,z);
        }

        if (ch==2)
        {
                cout<<"Give base1 and base2 of rectangle: ";
                cin>>x1>>y1;
                shapes s(x1,y1);
        }

        if (ch==3)
        {
                cout<<"Give base1 of square: ";
                cin>>x1;
                shapes s(x1);
        }

        if (ch==4)
        {
                cout<<"Give coordinates of 1st Point: ";
                cin>>x1>>y1;
                cout<<"Give coordinates of 2nd Point: ";
                cin>>x2>>y2;

                shapes s(x1,x2,y1,y2);
        }
```

```
        if (ch==5)
                shapes s;
}
```

12 Κληρονομικότητα

12.1 Απλή Κληρονομικότητα (inheritance)

Ένα από τα δυναμικότερα χαρακτηριστικά του αντικειμενοστραφούς προγραμματισμού είναι η κληρονομικότητα (inheritance). Τα αντικείμενα μιας κλάσης μπορεί να κληρονομούν μέλη δεδομένα και συναρτήσεις μιας άλλης κλάσης. Έτσι τις δυνατότητες των αντικειμένων μιας κλάσης μπορούμε να τις επαυξάνουμε δημιουργώντας κλάσεις με επιπρόσθετα επιθυμητά μέλη δεδομένα και συναρτήσεις. Η δημιουργία νέων αντικειμένων μπορεί να στηριχθεί σε ιδιότητες κλάσεων που ήδη έχουν δημιουργηθεί, έχουν ελεγχθεί χρησιμοποιούνται και προσφέρονται σε βιβλιοθήκες στην αγορά λογισμικού.

12.2 Βάση, Παραγόμενη και Άδειες Προσπέλασης

Η κλάση (γενικότερα ένας σύνθετος τύπος-δομή, κλάση, ένωση) που κληρονομείται σε κάποια άλλη λέγεται βάση (base), ενώ εκείνη που κληρονομεί ονομάζεται παραγόμενη (derived) κλάση. Όπως έχει αναφερθεί τα μέλη μιας κλάσης χαρακτηρίζονται με άδειες προσπέλασης ιδιωτικά (***private***:), προστατευμένα (***protected***:) και δημόσια (***publc***:).

Από την παραγόμενη κλάση επιτρέπεται η προσπέλαση των μελών της βάσης τα οποία είναι προστατευμένα (***protected***) ή δημόσια (***publc***). Τα ιδιωτικά (***private***) μέλη της βάσης είναι απροσπέλαστα από την παραγόμενη κλάση. Τα ιδιωτικά (***private***) μέλη μιας κλάσης χαρακτηρίζονται από το ότι, δεν μπορούν να προσπελαστούν παρά μόνο μέσα στον ορισμό συναρτήσεων, που είναι μέλη της κλάσης ή είναι φιλικές προς την κλάση. Τα προστατευμένα μέλη έχουν αυτή την ιδιότητα με μια εξαίρεση :"Μπορεί να προσπελαστούν και μέσα σε ορισμούς συναρτήσεων σε κλάσεις που είναι παραγόμενες". Εάν λοιπόν θέλουμε μια κλάση να αποκρύπτει ένα μέλος έξω από τον ορισμό της, θα δηλώσουμε αυτό το μέλος ιδιωτικό (***private***). Εάν όμως θέλουμε το μέλος ν α μπορεί να προσπελαύνεται και στον ορισμό (και μόνο) παραγόμενων από αυτήν την κλάσεων θα δηλώσουμε το μέλος προστατευμένο (***protected***). **Τόσο τα ιδιωτικά μέλη, όσο και τα προστατευμένα δεν προσπελαύνονται εκτός του ορισμού των κλάσεων (δηλαδή από τα αντικείμενα των κλάσεων). Τα δημόσια μέλη μιας κλάσης είναι προσπελάσιμα μέσα και έξω από τον ορισμό μιας κλάσης. Φυσικά μπορεί να προσπελαστούν και μέσα στον ορισμό μιας παραγόμενης κλάσης**.

12.3 Χαρακτηρισμοί Κληρονομικότητας

Μια κλάση γίνεται παραγόμενη κάποιας άλλης (της βάσης) με τρεις διαφορετικούς χαρακτηρισμούς κληρονομιάς: Η κληρονομιά μπορεί να είναι δημόσια (***publc***) ή προστατευμένη (***protected***) ή ιδιωτική (***private***). Οι χαρακτηρισμοί κληρονομιάς προσδιορίζουν τις άδειες προσπέλασης των μελών που κληρονομούνται από την βάση στην παραγόμενη κλάση.

Εάν η κληρονομιά είναι **δημόσια**, τότε τα μέλη της βάσης που είναι:

α)**δημόσια**, θα είναι για την παραγόμενη κλάση **δημόσια μέλη**

β)**προστατευμένα**, θα είναι για την παραγόμενη κλάση **προστατευμένα μέλη**

γ)**ιδιωτικά**, θα είναι στην παραγόμενη κλάση **απροσπέλαστα**.

Εάν η κληρονομιά είναι **προστατευμένη**, τότε τα μέλη της βάσης, που είναι:

α)**δημόσια**, θα είναι για την παραγόμενη κλάση **προστατευμένα μέλη**

β)**προστατευμένα**, θα είναι για την παραγόμενη κλάση **προστατευμένα μέλη**

γ)**ιδιωτικά**, θα είναι στην παραγόμενη κλάση **απροσπέλαστα**.

Εάν η κληρονομιά είναι **ιδιωτική**, τότε τα μέλη της βάσης, που είναι:

α)**δημόσια**, θα είναι για την παραγόμενη κλάση **ιδιωτικά μέλη**

β)**προστατευμένα**, θα είναι για την παραγόμενη κλάση **ιδιωτικά μέλη**

γ)**ιδιωτικά**, θα είναι στην παραγόμενη κλάση **απροσπέλαστα**.

Ο επόμενος πίνακας δίνει μια σύνοψη του πως τροποποιούνται οι άδειες προσπέλασης των μελών μιας βάσης σε παραγόμενη κλάση ανάλογα με τον χαρακτηρισμό της κληρονομιάς. Ο επόμενος πίνακας δίνει μια σύνοψη του πως τροποποιούνται οι άδειες προσπέλασης των μελών μιας βάσης σε παραγόμενη κλάση, ανάλογα με τον χαρακτηρισμό της κληρονομιάς.

Σε παραγόμενη κλάση με κληρονομιά:	**Public: μέλος (βάσης γίνεται:)**	**Protected: μέλος (βάσης γίνεται:)**	**Private: μέλος (βάσης γίνεται:)**
publc	***publc***:	***protected***:	απροσπέλαστο
protected	***protected***:	***protected***	απροσπέλαστο
private	***private***:	***private***	απροσπέλαστο

12.4 Γιατί και πως χρησιμοποιείται η Κληρονομικότητα

Κληρονομικότητα χρησιμοποιούμε βασικά όταν θέλουμε να τροποποιήσουμε τον κώδικα ενός σύνθετου τύπου. Η δυνατότητα αυτή μας επιτρέπει και "κτίσιμο" καλύτερα δομημένου προγράμματος, διότι από μικρούς σύνθετους τύπους χωρίς να ενδιαφερόμαστε για λεπτομέρειες δημιουργούμε πιο δυναμικούς σύνθετους τύπους.

Παράδειγμα 1 - Πρόγραμμα με κληρονομικότητα τύπου Public

```
#include <iostream.h>
#include <stdlib.h>
```

```
class shapes//Αρχική κλάση ή κλάση βάσης.
{
pu1blic:
        shapes()
        {
            cout<<"constructor of shapes called"<<endl;
            cout<<"give values of length,width,height "<<endl;
            cin>>length>>width>>height;
        }

protected://Ολα τα μέλh μιας κλάσης που δηλώνονται ως protected μπορούν να προσπελασθούν είτε μέσα στην κλάση στην οποία ορίζονται είτε στις παραγόμενες κλάσεις και πουθενά αλλού.
        double length,width,height,area;

        void print_area()
        {
            cout<<"the area is "<<area<<endl;
        }
};

class square:publc shapes //Αυτό σημαίνει ότι η κλάση square κληρονομείται από την κλάση shapes με κληρονομικότητα τύπου publc.Αυτό σημαίνει οτι:τα publc μέλη της shapes κληρονομούνται ως publc, τα protected ως protected και τα private καθόλου
{
publc:
        square()
        {
            cout<<"constructor of square called"<<endl;
        }

        void set_area_sq()
        {
            area=length*length;//η length και η area κληρονομούνται από την shapes
        }

        void print_area_sq()
        {
            print_area();//κληρονομείται από την shapes
        }
};

class rectangle:publc shapes //Αυτό σημάινει οτι η κλάση rectangle κληρονομείται από την κλάση shapes με κληρονομικότητα τύπου publc.Αυτό σημαίνει οτι:τα publc μέλη της shapes κληρονομούνται ως publc, τα protected ως protected και τα private καθόλου
{
        publc:
        rectangle()
        {
                cout<<"constructor of rectangle called"<<endl;
```

```
	}

	void set_area_rec()
	{
		area=length*width;//η length, η width και η area κληρονομούνται από την shapes
	}

	void print_area_rec()
	{
		print_area();//κληρονομέιται από τη shpes
	}
};

class trapezoid:publc shapes //Αυτό σημαίνει ότι η κλάση trapezoid κληρονομείται από την κλάση shapes με κληρονομικότητα τύπου publc.Αυτό σημαίνει ότι:τα publc μέλη της shapes κληρονομούνται ως publc, τα protected ως protected και τα private καθόλου
{
	publc:
	trapezoid()
	{
		cout<<"constructor of trapezoid called"<<endl;
	}

	void set_area_trap()
	{
		area=(length*width)*height/2;//η length, η width, η height και η area κληρονομούνται από την shapes
	}

	void print_area_trap()
	{
		print_area();//κληρονομείται από τη shapes
	}
};

void main()
{
	int ch;
	char ans;

	do
	{
	cout<<"1-area of square"<<endl;
	cout<<"2-area of rectangle "<<endl;
	cout<<"3-area of trapezoid "<<endl;
	cout<<"4-exit "<<endl;

	cout<<"choose "<<endl;
	cin>>ch;
```

```
        switch(ch)
        {
        case 1:{
                                square s //Καλείται αυτομάτως ο constructor της κλάσης square. Επειδή όμως η square κληρονομείται από τη shapes θα κληθεί αυτομάτως και ο constructor της shapes
                                s.set_area_sq();
                                s.print_area_sq();
                                break;
                }

        case 2: {
                                rectangle r;
                                r.set_area_rec();
                                r.print_area_rec();
                                break;
                        }

        case 3:{        trapezoid t;
                                t.set_area_trap();
                                t.print_area_trap();
                                break;
                }

        case 4: exit(1);
        }

        cout<<"continue? (y/n) "<<endl;
        cin>>ans;
        system("cls");
        }
        while (ans='y');
}
```

<u>Παράδειγμα 2 - Πρόγραμμα με προστατευμένη (protected) κληρονομικότητα</u>

```
#include <iostream.h>
//Β! τρόπος κληρονομικότητας: Προστατευμένη Κληρονομικότητα
//Τα publc μέλη κληρονομούνται σαν protected, τα protected σαν protected και τα private σαν απροσπέλαστα
class A
{
        int x;//Το μέλος x εννοείται ότι είναι private

protected: int y;

publc:  int z;
                void get(){cout<<" Δώσε x,y,z :";cin>>x>>y>>z;}
```

```
        void show(){cout<<"x="<<x<<" y="<<y<<" z="<<z<<'\n';}
};

class B:protected A{};//Η κλάση Β δεν έχει δικά της μέλη, αλλά κληρονομεί τα μέλη της κλάσης Α

void main()
{
B b;//Ορίζεται αντικείμενο της κλάσης Β
//b.get(); αυτό είναι λάθος διότι το αντικείμενο b δεν μπορεί να προσπελάσει τη συνάρτηση get() αφού αυτή κληρονομείται σαν protected από τη βάση
//b.show(); αυτό είναι λάθος διότι το αντικείμενο b δεν μπορεί να προσπελάσει τη συνάρτηση show() αφού αυτή κληρονομείται σαν protected από τη βάση
//b.z=2; αυτό είναι λάθος διότι το αντικείμενο b δεν μπορεί να προσπελάσει τη μεταβλητή z αφού αυτή κληρονομείται σαν protected από τη βάση
    //cout<<b.z<<endl; αυτό είναι λάθος διότι το αντικείμενο b δεν μπορεί να προσπελάσει τη μεταβλητή z αφού αυτή κληρονομείται σαν protected από τη βάση
    //b.y=1; αυτό είναι λάθος διότι το y είναι protected μέλος της βάσης άρα δεν μπορεί να προσπελαστεί έξω από τον ορισμό της κλάσης βάσης ή της παραγόμενης κλάσης
    //b.x=0; αυτό είναι λάθος διότι το x είναι private μέλος της βάσης και κληρονομείται σαν απροσπέλαστο
    //Κάθε παραγόμενη κλάση με βάση την κλάση Β δεν έχει προσπέλαση στο μέλος x

//Δηλαδή ουσιαστικά το πρόγραμμα αυτό δεν κάνει τίποτα

//Στην προστατευμένη κληρονομικότητα στην παραγόμενη κλάση δεν μπορούν να προσπελαστούν τα publc και τα protected μέλη της βάσης έξω από την παραγόμενη κλάση αλλά ούτε και τα private μέλη της βάσης

//Τα private μέλη μιας κλάσης προσπελαύνονται μόνο στον ορισμό της κλάσης και πουθενά αλλού
//Τα protected μέλη μιας κλάσης προσπελαύνονται και στον ορισμό της κλάσης αλλά και στον ορισμό των παραγόμενων κλάσεων
//Τα publc μέλη μιας κλάσης είναι προσπελάσιμα και στον ορισμό της κλάσης αλλά και έξω από τον ορισμό της κλάσης δηλαδή παντού
}
```

12.5 Κατασκευαστές και Καταστροφέας σε Κληρονομιά

Δημιουργώντας ένα αντικείμενο μιας παραγόμενης (derived) κλάσης, τότε λειτουργεί αρχικά ο κατασκευαστής (constructor) που είναι προσδιορισμένος για την κλάση βάση και στη συνέχεια ο κατασκευαστής, που είναι προσδιορισμένος για την παραγόμενη κλάση. Αν δεν έχουμε δικούς μας κατασκευαστές, τότε λειτουργούν οι καθιερωμένοι (default).

Όταν καταστρέφεται ένα αντικείμενο μιας παραγόμενης κλάσης, τότε λειτουργεί αρχικά ο καταστροφέας (destructor) της παραγόμενης κλάσης και στη συνέχεια ο καταστροφέας της βάσης (και πάλι αν δεν έχουμε ορίσει δικούς μας καταστροφείς λειτουργούν οι καθιερωμένοι).

Παράδειγμα 1 - Constructor & Destructor σε κληρονομικότητα

```
#include <iostream.h>
```

```
class A
{
protected:
        A()
        {
                cout<<"Object of class A"<<endl;
        }
        ~A()
        {
                cout<<"Destructor of object of class B"<<endl;
        }
};


class B:publc A
{
publc:
        B()
        {

                cout<<"Object of class B"<<endl;
        }
        ~B()
        {
                cout<<"Destructor of object of class B"<<endl;
        }
};

void main()
{
        B b;
}
```

12.6 Υπερφόρτωση (overloading) Τελεστών

Οι τελεστές στη C++ είναι σύμβολα που αντιπροσωπεύουν ειδικές συναρτήσεις οι οποίες ονομάζονται **συναρτήσεις τελεστές** (operator functions). Χρησιμοποιώντας τον τελεστή + μπορούμε να προσθέσουμε δύο μεταβλητές a, b με την παράσταση a+b. Στην πραγματικότητα καλούμε την συνάρτηση που αντιστοιχίζεται στον τελεστή + με παραμέτρους τους τελεστέους (operands) a, b. Γνωρίζουμε ότι την ίδια παράσταση a+b χρησιμοποιούμε για την πρόσθεση είτε οι τελεστέοι είναι μεταβλητές τύπου ***int*** είτε είναι τύπου ***double***. Αυτό συμβαίνει διότι στη συνάρτηση τελεστή + έχουν γίνει κατάλληλες υπερφορτώσεις (overloadings). Στη C++ έχουμε δυνατότητα να χρησιμοποιούμε τελεστές σε αντικείμενα ενός σύνθετου τύπου αρκεί να υπερφορτώσουμε (overload) τις αντίστοιχες συναρτήσεις τελεστές. Για παράδειγμα επιτρέπεται να υπερφορτώσουμε τον τελεστή + σε μια κλάση ώστε να εκτελεί την πρόθεση δύο αντικειμένων της κλάσης. Το αποτέλεσμα της πράξης το προκαθορίζουμε ανάλογα με τις εντολές που δίνουμε στην υπερφόρτωση της συνάρτησης τελεστή +. Τις συναρτήσεις τελεστές που υπερφορτώνουμε για αντικείμενα κάποιου σύνθετου τύπου (δομή, ένωση κλάση) τις χειριζόμαστε σαν συναρτήσεις που έχουν την μορφή:

τύπος operator σύμβολο (παράμετροι)

Για παράδειγμα αν έχουμε δύο αντικείμενα a, b της κλάσης A και θέλουμε η παράσταση a+b να υπολογίζει ένα ακέραιο που προκύπτει από κάποιες πράξεις σε μέλη των αντικειμένων αρκεί να προσδιορίσουμε μια συνάρτηση τελεστή της οποίας το πρωτότυπο θα έχει την μορφή:

int **operator + (A a, A b);**

12.7 Τρόποι Υπερφόρτωσης των Τελεστών

Η υπερφόρτωση των τελεστών για τα αντικείμενα ενός σύνθετου τύπου μπορεί να γίνει κατά κανόνα με δύο τρόπους:

1. **Ορίζοντας συναρτήσεις τελεστές μέλη του τύπου**
2. **Ορίζοντας φιλικές προς τον τύπο συναρτήσεις τελεστές.**

Οι τελεστές που μπορεί να υπερφορτωθούν έχουν ένα ή δύο τελεστέους και ανάλογα με την περίπτωση θα πρέπει να ορισθούν συναρτήσεις τελεστές με προσδιορισμό μιας ή δύο παραμέτρων [εξαίρεση παρουσιάζεται στον τελεστή ()]. Για παράδειγμα ο τελεστής + έχει δύο τελεστέους και ο τελεστής ++ έχει ένα τελεστέο. Επομένως η υπερφόρτωση του τελεστή + θα πρέπει να γίνει με προσδιορισμό δύο παραμέτρων ενώ η υπερφόρτωση του τελεστή ++ θα πρέπει να γίνει με προσδιορισμό μιας παραμέτρου. Εάν για την υπερφόρτωση τελεστή χρησιμοποιήσουμε συνάρτηση τελεστή μέλος θα πρέπει να έχουμε υπόψη ότι η πρώτη παράμετρος είναι ήδη προσδιορισμένη και δεν πρέπει να αναφερθεί στον ορισμό. Θεωρείται το αντικείμενο που είναι αριστερά του τελεστή (αν έχει δύο τελεστέους) ή το αντικείμενο επί του οποίου ορίζεται η πράξη (αν έχει ένα τελεστέο) **είναι το αντικείμενο *this**. Έτσι χρησιμοποιώντας υπερφόρτωση με συνάρτηση τελεστή μέλος το πρωτότυπο για την υπερφόρτωση του τελεστή + θα έχει μόνο μία παράμετρο (την δεύτερη) δηλαδή θα έχει την μορφή:

τύπος operator + (παράμετρος_δεξιά);

ενώ το πρωτότυπο για την υπερφόρτωση του ++ δεν θα έχει παράμετρο δηλαδή θα έχει την μορφή:

τύπος operator ++();

Χρησιμοποιώντας φιλική συνάρτηση για την υπερφόρτωση θα πρέπει να δώσουμε αντίστοιχα για τον τελεστή + δύο παραμέτρους (η σειρά δεν παίζει ρόλο), δηλαδή θα έχουμε την μορφή:

τύπος operator +(1η_παράμετρος, 2η_παράμετρος);

και για τον τελεστή ++ θα πρέπει να δώσουμε μια παράμετρο με την μορφή:

τύπος operator ++(παράμετρος);

13 Τελεστές που Υπερφορτώνονται και Περιορισμοί

Η C++ μας επιτρέπει να υπερφορτώσουμε όλους τους γνωστούς τελεστές εκτός των επόμενων 5 τελεστών:

- **::**(τελεστής εύρους)
- **.**(τελεία-προσπέλαση μέλους)
- **.***(τελεία_αστέρι-προσπέλαση μέλους με pointer προς το μέλος)
- **?**(τριαδικός τελεστής)
- **sizeof** (υπολογίζει πλήθος bytes).

Κατά την υπερφόρτωση των τελεστών πρέπει να λάβουμε υπόψη μας τα ακόλουθα:

- **Δεν είναι δυνατόν να δημιουργήσουμε τελεστές με δικά μας σύμβολα**. Το σύμβολο @ δεν είναι ορισμένος τελεστής στη C++ άρα δεν μπορούμε να χρησιμοποιήσουμε τέτοιο τελεστή. Ειδικά σε αντικείμενα μπορούμε να χρησιμοποιήσουμε τον τελεστή ^ υπερφορτώνοντας την αντίστοιχη συνάρτηση τελεστή αφού είναι ορισμένος στην C++.
- Οι υπερφορτωμένοι τελεστές **διατηρούν την βαθμίδα προτεραιότητας** που έχουν και δεν υπάρχει τρόπος να τη μεταβάλλουμε (η πράξη /όπως και να ορίζεται για τα αντικείμενα σε μια παράσταση χωρίς παρενθέσεις θα γίνεται πριν την πράξη +)
- Δεν επιτρέπεται να **αλλάξουμε το πλήθος των τελεστέων** (operands). (ο τελεστής ! ανεξάρτητα του τι θα ορίσουμε να κάνει, θα δέχεται ένα τελεστέο και ο τελεστής +πάντοτε δύο). Εξαίρεση έχουμε στον τελεστή ().
- Στην υπερφόρτωση της συνάρτησης τελεστή **δεν επιτρέπονται καθιερωμένες (default) παράμετροι** (π.χ. όχι δεκτό ***int*** operator + (A a, ***int*** k=0))

Οι επόμενοι 4 τελεστές υπερφορτώνονται μόνο με συναρτήσεις μέλη (Στον τελεστή () δεν έχουμε περιορισμό για το πλήθος των παραμέτρων)

- [] (υπολογισμός στοιχείου πίνακα),
- () (υπολογισμός τιμής συνάρτησης),
- .(προσπέλαση μέλους με pointer) και
- =(εκχώρηση τιμής assignment).

Γενικά υπερφορτώνοντας ένα τελεστή προκειμένου να τον χρησιμοποιήσουμε σε αντικείμενα μπορούμε να προσδιορίσουμε ελεύθερα την πράξη που θα επιτελέσει. Για παράδειγμα να χρησιμοποιούμε το σύμβολο + για να εκτελούμε αφαίρεση ή να παίρνουμε μόνιμα το αποτέλεσμα 500. Όμως η υπερφόρτωση των τελεστών γίνεται με στόχο να

απλοποιούμε τις παραστάσεις στο πρόγραμμα μας και όχι να περιπλέξουμε το πρόγραμμα. Έτσι θα πρέπει να υπερφορτώνουμε τους τελεστές διατηρώντας όσο το δυνατόν μια λογικά παραδεκτή συμπεριφορά. Η υπερφόρτωση ορισμένων τελεστών σε αντικείμενα θα πρέπει να αποφεύγεται με την έννοια ότι η τροποποίηση δεν επιβάλλεται από ουσιαστικές ανάγκες προγραμματισμού εκτός από την δημιουργία εκτυπώσεων π.χ δυσκολεύεται κανείς να συλλάβει παράδειγμα που επιβάλλει την υπερφόρτωση του τελεστή &.

13.1 Η Κλάση car ως Παράδειγμα Υπερφόρτωσης

Υποθέτουμε ότι τα αντικείμενα της κλάσης car θα διαθέτουν δύο ιδιωτικά μέλη:
α)τη συμβολοσειρά (string) name 79 το πολύ χαρακτήρων στην οποία καταγράφεται η μάρκα αυτοκινήτου
β)την ακέραια μεταβλητή velocity στην οποία θα τοποθετείται η ταχύτητα του αυτοκινήτου

Ακόμη υποθέτουμε ότι η κλάση car διαθέτει δημόσιες συναρτήσεις μέλη κατασκευαστή (constructor) με τον οποίον μπορούμε να δημιουργούμε αντικείμενα:
α)με αρχική τιμή μόνο τη μάρκα οπότε η ταχύτητα θα παίρνει αυτόματα τιμή μηδέν και
β)με αρχικές τιμές σε μάρκα και σε ταχύτητα

Δηλαδή ο κατασκευαστής θα έχει καθιερωμένη (default) τιμή παραμέτρου για την ταχύτητα και τη συνάρτηση μέλους show() με την οποία θα εμφανίζονται οι τιμές των μελών name και velocity. Είναι γνωστό ότι δημιουργώντας μια κλάση, η C++ μας εφοδιάζει με ένα καθιερωμένο τελεστή =, που μπορούμε να χρησιμοποιήσουμε για τα αντικείμενα της κλάσης. Ο καθιερωμένος τελεστής = είναι δυαδικός τελεστής (δύο τελεστέοι operands) και η ισότητα a=b για τα αντικείμενα a, b προκαλεί την αντιγραφή των μελών δεδομένων του b στα αντίστοιχα μέλη του a.

13.2 Σύνοψη Χρήσης του Συμβόλου = με Αντικείμενα

Το σύμβολο = χρησιμοποιείται σε κλάσεις με δύο τρόπους:
1)Σαν διαχωριστής (separator ή punctuator) κατά τη δημιουργία νέου αντικειμένου με αρχική τιμή άλλο αντικείμενο όπως:
όνομα_κλάσης νέο_αντικείμενο = όνομα_κλάσης (αντικείμενο_υπάρχον);
Στην περίπτωση αυτή διαχωρίζει τον ορισμό του αντικειμένου από την αρχική τιμή.

2)Σαν τελεστής με δύο διακριτές χρήσεις:
α)εκχώρηση τιμής αντικειμένου σε αντικείμενο όπως
αντικείμενο=αντικείμενο; ή αντικείμενο=κλάση (αντικείμενο);
β)εκχώρηση τιμής κάποιου μέλους σε αντικείμενο όπως
αντικείμενο=τιμή; ή αντικείμενο =κλάση (τιμή);

13.3 Υπερφόρτωση των +(πρόσθεση), -(αφαίρεση) +(πρόσημο), -(πρόσημο)

Στην κλάση car (όπως και σε κάθε νέα κλάση) δεν υπάρχουν καθιερωμένοι τελεστές + ή – (όπως ισχύει για τον τελεστή =). Επομένως για τα αντικείμενα της κλάσης a, b οι παραστάσεις a+b; a-b; +a; -a; θα προκαλέσουν σφάλμα εκτός και έχει γίνει υπερφόρτωση των τελεστών + (πρόσθεση και πρόσημο) και –(αφαίρεση και πρόσημο). Θα ορίσουμε υπερφόρτωση των τελεστών αποδίδοντας στους τελεστές πράξεις με τις ακόλουθες έννοιες:

- Με a+b; θα εννοούμε αντικείμενο τύπου car που έχει name μη προσδιορισμένο (ο pointer name θα έχει τιμή ***NULL*** (=0)) και velocity το άθροισμα των ταχυτήτων των αντικειμένων a και b (δηλαδή a.velocity+b.velocity)

· Με a-b; θα εννοούμε αντικείμενο τύπου car που έχει name μη προσδιορισμένο (ο pointer name θα έχει τιμή ***NULL*** (=0)) και velocity τη διαφορά των ταχυτήτων των αντικειμένων a και b (δηλαδή a.velocity-b.velocity).

· Με +a θα εννοούμε τροποποίηση του αντικειμένου a ως προς την ταχύτητα. Η ταχύτητα θα παίρνει την απόλυτη τιμή της ταχύτητας που είχε (δηλαδή a.velocity=abs(a.velocity)).

· Με –a θα εννοούμε τροποποίηση του αντικειμένου a ως προς την ταχύτητα. Η ταχύτητα θα παίρνει αντίθετη τιμή (δηλαδή a.velocity=-a.velocity).

Η υπερφόρτωση των τελεστών + και - μπορεί να γίνει είτε με συναρτήσεις τελεστές μέλη είτε με φιλικές συναρτήσεις.

13.4 Υπερφόρτωση του Τελεστή <<

Το σύμβολο << ενώ είναι ένας και μοναδικός τελεστής χρησιμοποιείται στην C++ σαν τελεστής σε δύο διαφορετικές περιπτώσεις:

α)σαν ολίσθηση προς τα αριστερά και
β)σαν τελεστής εξόδου προς αντικείμενα που ανήκουν σε προκαθορισμένη από την C++ κλάση με όνομα ostream.

Εάν a, b είναι ακέραιοι τότε με a<<b; εκτελείται ο τελεστής ολίσθησης και παράγει αποτέλεσμα a*2b (b φορές ολίσθηση προς τα αριστερά του δυαδικού αριθμού a).

Αν a είναι αντικείμενο της κλάσης ostream και b μια σταθερά (ή μεταβλητή ή συμβολοσειρά), τότε με a<<b; γίνεται εξαγωγή (output) της τιμής του b υπό μορφή χαρακτήρων και εισαγωγή (insertion) των χαρακτήρων στο αρχείο με το οποίο έχει συνδεθεί το αντικείμενο a. Πράγματι με ***cout***<<15; εκτυπώνεται στην οθόνη το 15, διότι το ***cout*** είναι προκαθορισμένο αντικείμενο της κλάσης ostream και αποτελεί ρεύμα ροής (stream) προς τον στάνταρ μηχανισμό εξόδου (συνήθως οθόνη). Το αποτέλεσμα αυτό προκύπτει, διότι στην κλάση ostream ο τελεστής ολίσθηση έχει υπερφορτωθεί. Ο τελεστής εξόδου << (output) λέγεται και εισαγωγέας (inserter). Δημιουργώντας μια δική μας κλάση δεν έχουμε διαθέσιμο τον τελεστή << ούτε σαν αριστερή ολίσθηση (με πρώτο τελεστέο αντικείμενο της κλάσης και δεύτερο τελεστέο ακέραιο), ούτε σαν τελεστή εξόδου (με πρώτο τελεστέο αντικείμενο της ostream και δεύτερο τελεστέο αντικείμενο της κλάσης που δημιουργούμε). Για να επιτύχουμε κάτι τέτοιο θα πρέπει να υπερφορτώσουμε τον τελεστή <<ολίσθηση, ή τον τελεστή <<έξοδος προς ρεύμα ροής. Η υπερφόρτωση του τελεστή ολίσθησης μπορεί να γίνει, είτε με συνάρτηση μέλος, είτε με συνάρτηση φιλική κατά εντελώς ανάλογο τρόπο με αυτόν που εφαρμόσαμε για τους τελεστές +(άθροισμα) και –(διαφορά) π.χ. στην κλάση car ορίζοντας την υπερφόρτωση μέλους Car &operator <<(***int*** k) {velocity=velocity<<k; ***return*** *this;} μπορούμε να χρησιμοποιούμε την παράσταση a<<1; για να διπλασιάζουμε την ταχύτητα στο αντικείμενο a (της κλάσης car). Η έξοδος (output) προς ένα αντικείμενο της κλάσης ostream, όπως είναι το ***cout***, μπορεί να γίνει και πάλι με υπερφόρτωση του τελεστή << (εξόδου – output). Υπάρχουν για τον σκοπό αυτό δύο δυνατότητες:

· να επιχειρήσουμε να υπερφορτώσουμε τον τελεστή <<στην κλάση ostream (και να διαφοροποιήσουμε μια προκαθορισμένη από την C++ κλάση) και
· να υπερφορτώσουμε στην κλάση που δημιουργούμε τον τελεστή εξόδου <<.

Φυσικά είναι αρκετά τολμηρό (έως παράλογο!) να τροποποιούμε τα καθιερωμένα αρχεία της C++ για να εργαστούμε με κάποια κλάση που θα δημιουργήσουμε. Απομένει λοιπόν να υπερφορτώσουμε τον τελεστή εξόδου στην κλάση που δημιουργούμε.

13.5 Υπερφόρτωση του Τελεστή >>

Το σύμβολο >> είναι ένας και μοναδικός τελεστής που χρησιμοποιείται στην C++ σαν τελεστής σε δύο διαφορετικές περιπτώσεις.

α)σαν ολίσθηση προς τα δεξιά και
β)σαν τελεστής εισόδου προς αντικείμενα, που ανήκουν σε προκαθορισμένη από την C++ κλάση με όνομα istream.

Εάν a, b είναι ακέραιοι τότε με a>>b; εκτελείται ο τελεστής ολίσθηση και παράγει αποτέλεσμα a/2β; εκτελείται ο τελεστής ολίσθηση και παράγει αποτέλεσμα α/2b (b φορές ολίσθηση προς τα δεξιά του δυαδικού αριθμού a). Εάν a είναι αντικείμενο της κλάσης istream και b μια μεταβλητή, τότε με a>>b; γίνεται εισαγωγή (input) στο b κάποιας τιμής μετά από εξαγωγή από το αρχείο με το οποίο έχει συνδεθεί το αντικείμενο a. Πράγματι με cin>>b; εισάγουμε από το πληκτρολόγιο τιμή στο b, διότι το cin είναι προκαθορισμένο αντικείμενο της κλάσης istream και αποτελεί ρεύμα ροής (stream) για εξαγωγή από τα στάνταρ μηχανισμού εισόδου (κατά κανόνα το πληκτρολόγιο). Το αποτέλεσμα αυτό προκύπτει, διότι στην κλάση istream ο τελεστής ολίσθηση προς τα δεξιά έχει υπερφορτωθεί. Ο τελεστής εισόδου (input) >> λέγεται και εξαγωγέας (extractor). Δημιουργώντας μια δική μας κλάση δεν έχουμε διαθέσιμο τον τελεστή >> ούτε σαν δεξιά ολίσθηση (με πρώτο τελεστέο αντικείμενο της κλάσης και δεύτερο τελεστέο ακέραιο), αλλά ούτε και σαν τελεστή εισόδου (με πρώτο τελεστέο αντικείμενο της istream) και δεύτερο τελεστέο αντικείμενο της κλάσης που δημιουργούμε). Για να επιτύχουμε κάτι τέτοιο θα πρέπει να υπερφορτώσουμε, είτε τον ένα τελεστή >>ολίσθησης, είτε τον τελεστή >> είσοδος προς ρεύμα ροής. Η υπερφόρτωση του τελεστή ολίσθησης δεξιά μπορεί να γίνει, είτε με συνάρτηση μέλος, είτε με συνάρτηση φιλική κατά εντελώς ανάλογο τρόπο με αυτόν που εφαρμόσαμε για τον τελεστή << π.χ. στην κλάση car ορίζοντας την υπερφόρτωση μέλος: Car & operator >>(***int*** k) {velocity = velocity>>k; ***return*** *this;} μπορούμε να χρησιμοποιούμε την παράσταση a>>1; για να υποδιπλασιάζουμε την ταχύτητα στο αντικείμενο a (της κλάσης car). Η είσοδος από ένα αντικείμενο της κλάσης istream όπως είναι το cin, μπορεί να γίνει και πάλι με υπερφόρτωση του τελεστή >> (εισόδου –input). Υπάρχουν για τον σκοπό αυτό δύο δυνατότητες. Πρώτον να επιχειρήσουμε να υπερφορτώσουμε τον τελεστή >> στην κλάση istream (και να διαφοροποιήσουμε μια προκαθορισμένη από την C++ κλάση) και δεύτερον να υπερφορτώσουμε στην κλάση που δημιουργούμε τον τελεστή εισόδου>>. Φυσικά και πάλι η πλέον κατάλληλη διέξοδος είναι να υπερφορτώσουμε τον τελεστή εισόδου >> στην κλάση που δημιουργούμε. Ας υποθέσουμε ότι θέλουμε να επιτύχουμε υπερφόρτωση του τελεστή εισόδου στην κλάση car. Γνωρίζουμε ότι ο πρώτος τελεστέος στο τελεστή εξόδου >> είναι αντικείμενο της κλάσης istream. Επομένως για την υπερφόρτωση δεν μπορούμε να χρησιμοποιήσουμε συνάρτηση μέλος της car διότι τότε ο πρώτος τελεστέος θα ήταν (έμμεσα) το αντικείμενο *this (του τύπου car). Άρα θα χρησιμοποιήσουμε φιλική συνάρτηση. Στην υπερφόρτωση θα εισάγονται δύο παράμετροι ένα αντικείμενο a της istream και ένα αντικείμενο b της car και θα χρησιμοποιείται ο τελεστής εισόδου << για να διαβιβάζονται προς ρεύμα ροής a τα μέλη του αντικειμένου b (μεταβλητές βασικών τύπων). Για να διατηρήσουμε την ιδιότητα πολλαπλής εκτέλεσης του τελεστή >> θα πρέπει η συνάρτηση τελεστής >> να επιστρέφει το αντικείμενο a με ***return*** a; Δηλαδή η επιστρεφόμενη τιμή θα είναι μια αναφορά (reference) προς την παράμετρο εισόδου a. Αυτό επιβάλλει ότι και η παράμετρος a θα πρέπει να είναι αναφορά (διότι διαφορετικά θα είχαμε αναφορά προς

τοπική μεταβλητή). Από την εκτέλεση του τελεστή >> το αντικείμενο που εκφράζει η παράμετρος b θα πρέπει να τροποποιείται (αφού θα εισάγονται τιμές στα μέλη του). Επομένως και η παράμετρος b θα πρέπει να είναι μια αναφορά (reference) προς αντικείμενα του τύπου car.

13.6 Υπερφόρτωση του Τελεστή ++

Ο τελεστής ++ έχει διπλή συμπεριφορά στη C++. Παρουσιάζεται είτε σαν πρόθεμα (prefix) μιας μεταβλητής (οπότε αυξάνει την τιμή της μεταβλητής πριν επακολουθήσει κάποια πράξη στην οποία συμμετέχει η μεταβλητή) είτε σαν επίθεμα (postfix) (οπότε εκτελείται κάποια πράξη στην οποία συμμετέχει η μεταβλητή και στην συνέχεια αυξάνεται η τιμή της μεταβλητής) π.χ. με b=++a; πρώτα αυξάνεται το a και μετά εκτελείται η ισότητα (εκχωρείται τιμής), ενώ στην παράσταση b=a++; πρώτα γίνεται η εκχώρηση της τιμής του a στο b και μετά αυξάνεται η τιμή της a. Η διπλή αυτή συμπεριφορά του τελεστή ++ οφείλεται στο ότι υπάρχουν αφιερωμένες στον τελεστή δύο συναρτήσεις τελεστές. Η πρώτη αντιστοιχίζεται στην συμπεριφορά σαν πρόθεμα και παίρνει μια παράμετρο (την μεταβλητή που αυξάνει) ενώ η δεύτερη αντιστοιχίζεται στην συμπεριφορά σαν επίθεμα και παίρνει δύο παραμέτρους. Η πρώτη παράμετρος του τελεστή ++ σαν επίθεμα είναι η μεταβλητή που αυξάνει (αριστερά του τελεστή) και η δεύτερη είναι μια μεταβλητή ακέραιου τύπου η οποία δεν χρησιμοποιείται και μπορεί να παραληφθεί. Η δεύτερη παράμετρος στην συνάρτηση τελεστής ++ επίθεμα απλά χρησιμεύει για την διαφοροποίηση από την συνάρτηση τελεστής ++ πρόθεμα.

Στα αντικείμενα μιας κλάσης όπως η car υπερφορτώνοντας τον τελεστή ++ με μια παράμετρο είναι σαν να υπερφορτώνουμε την συμπεριφορά του τελεστή σαν πρόθεμα, ενώ υπερφορτώνοντας τον τελεστή με δύο παραμέτρους προσδιορίζουμε την συμπεριφορά του τελεστή ++ σαν επίθεμα. Οι υπερφορτώσεις του ++ μπορεί να γίνουν είτε με συναρτήσεις μέλη, είτε με φιλικές συναρτήσεις.

Στην υπερφόρτωση σαν πρόθεμα αυξάνουμε την τιμή της πρώτης παραμέτρους και επιστρέφουμε την πρώτη παράμετρο. Επομένως η πρώτη παράμετρος θα είναι αναφορά (reference) προς αντικείμενο και η επιστρεφόμενη τιμή θα είναι το ίδιο το αντικείμενο παράμετρος αφού τροποποιηθεί δηλαδή και πάλι αναφορά προς αντικείμενο.

Στην υπερφόρτωση σαν επίθεμα θα πρέπει να επιστρέψουμε ένα τοπικό αντικείμενο που έχει την τιμή την πρώτη παράμετρο και μετά να αυξήσουμε την τιμή της παραμέτρου. Επομένως μέσα στην συνάρτηση υπερφόρτωσης θα πρέπει να τροποποιήσουμε μόνιμα την πρώτη παράμετρο. Άρα η πρώτη παράμετρος θα είναι αναφορά. Η επιστρεφόμενη τιμή θα είναι το τοπικό αντικείμενο (όχι αναφορά).

13.7 Παραδείγματα υπερφόρτωσης τελεστών

Υπερφόρτωση του τελεστή =

```
#include <iostream.h>
#include <conio.h>
#include <string.h>

class car
{
char name[80];
int velocity;

public:
```

```
    car(char s[80],int v=0)
    {
       strcpy(name,s);
       velocity=v;
    }

    void show()
    {
       cout<<name<<" TAXYTHTA= "<<velocity<<'\n';
    }
};

void main()
{
    car a("volvo "),b("Fiat "),c("Renault ",40),d("Mazda ",100);//Ορίζονται τα αντικείμενα a,b,c,d

    a.show();
    b.show();
    c.show();
    d.show(); //Εμφανίζονται τα μέλη των a,b,c,d

    a=car(c); //Είναι το ίδιο με την καταχώριση a=c
    b=a;//Εμφανίζονται τα μέλη των a,b,c,d
    cout<<'\n';

    a.show();
    b.show();
    c.show();
    d.show(); //Εμφανίζονται τα μέλη των a,b,c,d
    cout<<'\n';

    a=b=c=d;//πολλαπλή καταχώριση
    a.show();
    b.show();
    c.show();
    d.show(); //Εμφανίζονται τα μέλη των a,b,c,d
}
```

Υπερφόρτωση του τελεστή = με συνάρτηση μέλος

```
#include <iostream.h>
#include <conio.h>
#include <string.h>

class car
{
char name[80];
int velocity;

public:
```

```
    car (char s[80], int v=0) //τίθεται το s στο name και το v στο velocity
    {
       strcpy(name,s);
       velocity=v;
    }

    void show()
    {
       cout<<name<<" TAXYTHTA ="<<velocity<<'\n';
    }

    car operator=(car b)//υπερφόρτωση συνάρτησης τελεστή =
    {
       velocity=b.velocity;//Με τον τελεστή καταχώρισης = τροποποιείται μόνο η velocity
       return *this;//επιστρέφεται αντίγραφο του τρέχοντος αντικειμένου
    }
};
```

2 Β Μέρος – Χειρισμός και Περιγραφή Πληροφοριακού Συστήματος

Στην ενότητα αυτή παρουσιάζουμε την κατασκευή ενός πληροφοριακού συστήματος σε C++ για την αυτοματοποίηση των λειτουργιών μιας εμπορικής εταιρίας η οποία διαχειρίζεται κάποια είδη, αγοράζει πρώτες ύλες από προμηθευτές και πραγματοποιεί πωλήσεις σε πελάτες της.. Το πληροφοριακό σύστημα στηρίζεται στις ακόλουθες κλάσεις:

- class Date **à** Το αρχείο περιέχει την κλάση περιγραφής των ημερομηνιών
- class ClientData **à** Η κλάση αυτή περιγράφει τα στοιχεία ενός πελάτη
- class SupplierData **à** Η κλάση αυτή περιγράφει τα στοιχεία ενός προμηθευτή
- class ItemData **à** Η κλάση αυτή περιγράφει τα στοιχεία ενός είδους
- class PaymentCustomerData **à** Η κλάση αυτή περιγράφει τα στοιχεία πληρωμής ενός πελάτη
- class PaymentSupplierData **à** Η κλάση αυτή περιγράφει τα στοιχεία πληρωμής ενός προμηθευτή
- class SaleCustomerData **à** Η κλάση αυτή περιγράφει τα στοιχεία πώλησης ενός πελάτη
- class SaleSupplierData **à** Η κλάση αυτή περιγράφει τα στοιχεία αγοράς από ένα προμηθευτή
- class Transaction **à** Η κλάση αυτή περιγράφει την κίνηση ενός πελάτη ή ενός προμηθευτή

Η βάση δεδομένων του πληροφοριακού συστήματος στηρίζεται σε 3 αρχεία

Items.txt **à** στο αρχείο αυτό καταχωρούνται όλα τα είδη

Customer.txt **à** στο αρχείο αυτό καταχωρούνται όλοι οι πελάτες

Supplier.txt **à** στο αρχείο αυτό καταχωρούνται όλοι οι προμηθευτές

Ο τρόπος υλοποίησης του προγράμματος είναι με τη δημιουργία headers files όπου γράφονται όλοι οι ορισμοί των κλάσεων και cpp files όπου υλοποιοούνται οι μέθοδοι των κλάσων. Τα αρχεία της εφαρμογής είναι τα ακόλουθα:

- transaction.h **à** header file με τον ορισμό της κλάσης Transaction
- item_functions.cpp **à** cpp file για την υλοποίηση μεθόδων που καλούνται στο main
- client_functions.cpp **à** cpp file για την υλοποίηση κάποιων μεθόδων που χρησιμοποιούνται στο main
- supplier_functions.cpp **à** cpp file για την υλοποίηση κάποιων μεθόδων που χρησιμοποιούνται στο main
- date.h **à** header file με τον ορισμό της κλάσης Date
- clients.h **à** header file με τον ορισμό της κλάσης ClientData
- clients.cpp **à** cpp file για την υλοποίηση των μεθόδων της κλάσης ClientData
- suppliers.h **à** header file με τον ορισμό της κλάσης SupplierData
- supplier.cpp **à** cpp file για την υλοποίηση των μεθόδων της κλάσης SupplierData
- items.h **à** header file με τον ορισμό της κλάσης ItemData
- items.cpp **à** cpp file για την υλοποίηση μεθόδων της κλάσης Item
- paymentcustomer.h **à** header file με τον ορισμό της κλάσης PaymentCustomerData
- paymentcustomer.cpp **à** cpp file για την υλοποίηση των μεθόδων της κλάσης PaymentCustomerData
- paymentsupplier.h **à** header file με τον ορισμό της κλάσης PaymentSupplierData
- paymentsupplier.cpp **à** cpp file για την υλοποίηση των μεθόδων της κλάσης PaymentSupplierData
- salecustomer.h **à** header file με τον ορισμό της κλάσης SaleCustomerData
- salecustomer.cpp **à** cpp file για την υλοποίηση των μεθόδων της κλάσης SaleCustomerData
- salesupplier.h **à** header file με τον ορισμό της κλάσης SaleSupplierData

- salesupplier.cpp **à** cpp file για την υλοποίηση των μεθόδων της κλάσης SaleSupplierData

2.1 Περιγραφή Λειτουργιών Πληροφοριακού Συστήματος

Ο χειρισμός του όλου προγράμματος αρχίζει από το αρχείο **menu.cpp**. Κατά την εκτέλεση του το αρχείο αυτό εμφανίζει το ακόλουθο μενού:

```
cout<<"1.Customer's Menu\n";

cout<<"2.Supplier's Menu\n";

cout<<"3.Item's menu\n";

cout<<"4.Program end\n";



cout<<"Choose: ";

cin>>ch0;
```

Εικόνα 1-Βασικό Μενού Χειρισμού Προγράμματος

13.7.1 Περιγραφή Λειτουργιών του υπομενού Πελατών

Επιλέγοντας 1 στο Βασικό Μενού Χειρισμού του Προγράμματος ο χρήστης έχει τη δυνατότητα να εργαστεί με τους πελάτες της επιχείρησης. Συγκεκριμένα με την επιλογή του Customer's Menu στο προηγούμενο μενού, εμφανίζεται στο χρήστη το ακόλουθο υπομενού:

```
cout<<"CUSTOMERS MENU\n";

cout<<"================\n";



cout<<"0.Delete file of customers\n";

cout<<"1.Insert New Customer\n";

cout<<"2.Print Customers\n";

cout<<"3.Update Customer\n";

cout<<"4.Delete Customer\n";

cout<<"5.Show balance of a specific Customer\n";
```

```
cout<<"6.Enter a new Customer's Sale\n";

cout<<"7.Print Customer's Sales\n"<<;

cout<<"8.Statistics Sales Per Customer (Sales#, Total and Average Amount)\n";

cout<<"9.Enter a Customer payment\n";

cout<<"10.Print Customer Payments\n";

cout<<"11.Return to main menu\n";



cout<<"Choose (1-11): ";

cin>>ch;
```

Εικόνα 2-Υπομενού Πελατών

Οι λειτουργίες αυτού του μενού περιγράφονται συνοπτικά στον ακόλουθο πίνακα:

Επιλογή	**Αποτέλεσμα**

0.Delete file of customers-payments and sales	Διαγράφεται το αρχείο πελατών με την εκτέλεση της συνάρτησης delete_clients(). Θα πρέπει να τονίσουμε ότι για τη διατήρηση της συνέπειας του συστήματος διαγράφονται ταυτόχρονα εκτός από τους πελάτες, οι πληρωμές και οι πωλήσεις τους με τις ακόλουθες δηλώσεις:
	
	ofstream acf("customer.txt");
	ofstream acf1("paymentcustomer.txt");
	ofstream acf2("salecustomer.txt");
	
	Αυτό γίνεται διότι δεν έχει νόημα να «φυλάγονται» οι πληρωμές και οι πωλήσεις για πελάτες που δεν υπάρχουν πλέον στο σύστημα

1.Insert New Customer	Στη λειτουργία αυτή καταχωρείται ένας νέος πελάτης στο σύστημα με την εκτέλεση της συνάρτησης insert_client(). Αρχικά ζητείται ο κωδικός του πελάτη και το σύστημα ελέγχει

	αν ο πελάτης αυτός υπάρχει ήδη καταχωρημένος οπότε τυπώνει μήνυμα λάθους και η καταχώριση τερματίζεται. Αν ο πελάτης δεν υπάρχει τότε ζητούνται τα υπόλοιπα στοιχεία του (όνομα, επώνυμο, διεύθυνση, ΤΚ, τηλέφωνο) και ο πελάτης καταχωρείται στο αρχείο customer.txt.
2.Print Customers	Με την επιλογή αυτή εμφανίζονται στην οθόνη όλοι οι πελάτες της εταιρείας που είναι καταχωρημένοι στο αρχείο customer.txt με την εκτέλεση της συνάρτησης print_clients()
3. Update Customer	Εδώ τροποποιείται το τηλέφωνο ή η διεύθυνση (ανάλογα με την επιλογή του χρήστη) ενός πελάτη της επιχείρησης με την εκτέλεση της συνάρτησης update_client(). Αρχικά ζητείται ο κωδικός του πελάτη και ελέγχεται η ύπαρξη του πελάτη. Αν ο πελάτης είναι όντως καταχωρημένος τότε ζητείται από το χρήστη να επιλέξει αν θέλει να τροποποιήσει το τηλέφωνο ή τη διεύθυνση του πελάτη και ανάλογα με την επιλογή του γίνεται η τροποποίηση.
4.Delete Customer	Με τη λειτουργία αυτή διαγράφεται ένας συγκεκριμένος πελάτης της επιχείρησης με την εκτέλεση της συνάρτησης delete_client(int cust). Το όρισμα cust της συνάρτησης είναι ο κωδικός του πελάτη που θέλουμε να διαγράψουμε. Πρώτα ελέγχουμε αν ο πελάτης είναι καταχωρημένος στο σύστημα και εφόσον αυτό συμβαίνει τότε το πρόγραμμα τον διαγράφει από το αρχείο customer.txt. Επίσης το πρόγραμμα διαγράφει και όλες τις πληρωμές του συγκεκριμένου πελάτη από το αρχείο paymentcustomer.txt καθώς και τις πωλήσεις του από το αρχείο salecustomer.txt.
5.Show balance of a specific Customer	Εδώ αρχικά ζητείται ο κωδικός του πελάτη και στη συνέχεια καλείται η συνάρτηση print_clients_balance(custbal) η οποία ανοίγει το αρχείο customer.txt, ελέγχει αν πελάτης υπάρχει καταχωρημένος σε αυτό και αν ναι τυπώνει το υπόλοιπο του αλλιώς εμφανίζει μήνυμα λάθους (δηλ. ότι ο πελάτης δεν υπάρχει). Με αυτή τη λειτουργία μπορούμε να ενημερωθούμε για το ποσό που οφείλει ένας πελάτης στην επιχείρηση.

6.Enter a new Customer's Sale	Η λειτουργία αυτή μας επιτρέπει να καταχωρήσουμε μια νέα πώληση ενός πελάτη στο αρχείο salecustomer.txt καλώντας τη συνάρτηση insert_salecustomer(). Αρχικά εισάγεται ο κωδικός του πελάτη και εφόσον αυτός είναι έγκυρος τότε εισάγεται ο κωδικός του είδους που θέλει να αγοράσει. Εφόσον και αυτός υπάρχει τότε ελέγχεται αν υπάρχει ικανό απόθεμα για να γίνει η πώληση στον πελάτη. Αν και το απόθεμα επαρκεί τότε πλέον καταχωρείται η πώληση.
7.Print Customer's Sales	Η λειτουργία αυτή εμφανίζει όλες τις πωλήσεις που έχει πραγματοποιήσει ένας συγκεκριμένος πελάτης εκτελώντας τη συνάρτηση print_client_sales(custom). Αρχικά εισάγεται πάλι ο κωδικός του πελάτη για τον οποίο θέλουμε να εμφανίσουμε τις πωλήσεις του, μετά ελέγχεται η εγκυρότητα του κωδικού αυτού και τέλος ανοίγει το αρχείο salecustomer.txt μέσα από το οποίο επιλέγονται και τυπώνονται οι πωλήσεις του συγκεκριμένου πελάτη.
8.Statistics Sales Per Customer (Sales#, Total and Average Amount)	Εδώ εμφανίζουμε κάποιες συγκεντρωτικές πληροφορίες για τους πελάτες μας με την εκτέλεση της συνάρτησης total_average_sales_per_ customer(). Πιο συγκεκριμένα χρησιμοποιώντας τις πληροφορίες του αρχείου salecustomer.txt υπολογίζουμε και εμφανίζουμε για τον κάθε πελάτη το συνολικό αριθμό των πωλήσεων που έχει πραγματοποιήσει, το συνολικό ποσό που έχει ξοδέψει γιαυτές και το μέσο ποσό ανά πώληση.
9.Enter a Customer payment	Με τη λειτουργία αυτή εισάγουμε-καταχωρούμε μια νέα πληρωμή ενός πελάτη στο αρχείο paymentcustomer.txt με την εκτέλεση της συνάρτησης insert_paymentcustomer(). Αρχικά γίνεται και πάλι ο έλεγχος εγκυρότητας για τον κωδικό του πελάτη και μετά εισάγονται η ημερομηνία (στην οποία γίνεται επίσης έλεγχος ορθής καταχώρισης) και το ποσό πληρωμής.
10.Print Customer Payments	Με την επιλογή αυτή εμφανίζουμε στην οθόνη τις πληρωμές όλων των πελατών. Η λειτουργία αυτή υλοποιείται με την εκτέλεση της συνάρτησης print_client_payments(custompp). Για κάθε πληρωμή εμφανίζονται τα στοιχεία του πελάτη, το ποσό

	της και η ημερομηνία που έγινε.
11.Return to main menu	Επιστρέφουμε στο βασικό μενού του προγράμματος που φαίνεται στην Εικόνα 1-Βασικό Μενού Χειρισμού Προγράμματος

13.7.2 Περιγραφή Λειτουργιών του υπομενού Προμηθευτών

Επιλέγοντας 2 στο Βασικό Μενού Χειρισμού του Προγράμματος ο χρήστης έχει τη δυνατότητα να εργαστεί με τους προμηθευτές της επιχείρησης. Συγκεκριμένα με την επιλογή του Supplier's Menu εμφανίζεται στο χρήστη το ακόλουθο υπομενού:

```
cout<<"SUPPLIERS MENU\n";

cout<<"================\n";



cout<<"0.Delete file of suppliers and payments and purchases\n";

cout<<"1.Insert New Supplier\n";

cout<<"2.Print Suppliers\n";

cout<<"3.Update Supplier\n";

cout<<"4.Delete Supplier\n";

cout<<"5.Show balance of a specific Supplier\n";

cout<<"6.Enter a new Supplier's Purchase"<<endl;

cout<<"7.Print Supplier's Purchases"<<endl;

cout<<"8.Statistics Per Supplier (Purchases#, Total and Average Amount)"<<endl;

cout<<"9.Enter a Supplier payment"<<endl;

cout<<"10.Print Supplier's Payments"<<endl;

cout<<"11.Return to main menu\n";



cout<<"Choose (1-11): ";

cin>>ch;
```

Οι λειτουργίες αυτού του υπομενού περιγράφονται στον ακόλουθο πίνακα:

Επιλογή	**Αποτέλεσμα**

0.Delete file of suppliers and payments and purchases	Διαγράφεται το αρχείο προμηθευτών, των αγορών που έχουμε πραγματοποιήσει από αυτούς και των πληρωμών που έχουμε κάνει σε αυτούς με την εκτέλεση της συνάρτησης delete_suppliers(). Στη συνάρτηση αυτή οι διαγραφές των αρχείων supplier.txt, paymentsupplier.txt και salesupplier.txt γίνονται με τις ακόλουθες δηλώσεις:
	
	ofstream acf("supplier.txt");
	ofstream acf1("paymentsupplier.txt");
	ofstream acf2("salesupplier.txt");
	
	Αυτό γίνεται διότι δεν έχει νόημα να «φυλάγονται» οι πληρωμές και οι αγορές από προμηθευτές που δεν υπάρχουν καταχωρημένοι πλέον στο σύστημα

1.Insert New Supplier	Με τη λειτουργία αυτή καταχωρείται ένας νέος προμηθευτής στο σύστημα. Αυτό γίνεται με την εκτέλεση της συνάρτησης insert_supplier(). Κάθε νέος προμηθευτής καταχωρείται στο αρχείο supplier.txt.
2.Print Suppliers	Με την επιλογή αυτή εμφανίζονται στην οθόνη όλοι οι προμηθευτές της εταιρείας που είναι καταχωρημένοι στο αρχείο suppliers.txt. Αυτό γίνεται με την εκτέλεση της συνάρτησης print_suppliers()
3. Update Supplier	Εδώ τροποποιείται το τηλέφωνο ή η διεύθυνση (ανάλογα με την επιλογή του χρήστη) ενός προμηθευτή της επιχείρησης με την εκτέλεση της συνάρτησης update_supplier(supp). Αρχικά ζητείται ο κωδικός του προμηθευτή και ελέγχεται αν αυτός καταχωρημένος στο αρχείο προμηθευτών (supplier.txt). Αν ναι τότε ζητείται από το χρήστη να επιλέξει αν θέλει να τροποποιήσει το τηλέφωνο ή τη διεύθυνση του προμηθευτή και ανάλογα με την επιλογή του γίνεται η τροποποίηση.
4.Delete Supplier	Με τη λειτουργία αυτή διαγράφεται ένας συγκεκριμένος προμηθευτής

	της επιχείρησης με την εκτέλεση της συνάρτησης delete_supplier (delsupp). Το όρισμα delsupp της συνάρτησης είναι ο κωδικός του προμηθευτή που θέλουμε να διαγράψουμε. Πρώτα ελέγχουμε αν ο προμηθευτής είναι καταχωρημένος στο σύστημα και εφόσον αυτό συμβαίνει τότε το πρόγραμμα τον διαγράφει από το αρχείο supplier.txt. Επίσης το πρόγραμμα διαγράφει και όλες τις πληρωμές που έχουν γίνει στο συγκεκριμένο προμηθευτή από το αρχείο paymentsupplier.txt καθώς και τις αγορές από αυτόν που είναι καταχωρημένες στο αρχείο salesupplier.txt.
5.Show balance of a specific Supplier	Στη λειτουργία αυτή ζητείται ο κωδικός του προμηθευτή και στη συνέχεια καλείται η συνάρτηση print_suppliers_balance(suppbal) η οποία ανοίγει το αρχείο supplier.txt, ελέγχει αν ο προμηθευτής είναι καταχωρημένος σε αυτό και αν ναι τυπώνει το υπόλοιπο που οφείλουμε σε αυτόν αλλιώς εμφανίζει το μήνυμα ότι ο προμηθευτής δεν υπάρχει). Με αυτή τη λειτουργία μπορούμε να ενημερωθούμε για το ποσό που οφείλουμε σε ένα προμηθευτή της επιχείρησης.
6.Enter a new Supplier's Purchase	Η λειτουργία αυτή μας επιτρέπει να καταχωρήσουμε μια νέα αγορά από κάποιο προμηθευτή της επιχείρησης καλώντας τη συνάρτηση insert_salesupplier(). Αρχικά εισάγεται ο κωδικός του προμηθευτή και εφόσον αυτός υπάρχει τότε εισάγεται ο κωδικός του είδους που θέλουμε να αγοράσουμε από αυτόν. Εφόσον και αυτός υπάρχει τότε ζητούνται η ποσότητα και η ημερομηνία αγοράς. Στην ημερομηνία γίνεται επίσης έλεγχος ορθής καταχώρισης.
7.Print Supplier's Purchases	Η λειτουργία αυτή εμφανίζει όλες τις αγορές που έχουμε πραγματοποιήσει από ένα συγκεκριμένο προμηθευτή με την εκτέλεση της συνάρτησης print_supplier_sales(suppl). Αρχικά ζητείται ο κωδικός του προμηθευτή, μετά ελέγχεται η ύπαρξη του κωδικού αυτού και τέλος ανοίγει το αρχείο salesupplier.txt μέσα από το οποίο επιλέγονται και τυπώνονται οι αγορές από το συγκεκριμένο προμηθευτή.
8.Statistics Per Supplier (Purchases#, Total and Average Amount)	Εδώ εμφανίζουμε στατιστικές πληροφορίες για τους προμηθευτές της επιχείρησης μας με την εκτέλεση της συνάρτησης total_average_sales_per_supplier(). Πιο συγκεκριμένα

	χρησιμοποιώντας τις πληροφορίες του αρχείου salesupplier.txt εμφανίζουμε για κάθε προμηθευτή το πλήθος των αγορών που έχουμε κάνει από αυτόν, το συνολικό ποσό που έχουμε ξοδέψει γιαυτές και το μέσο όρο της αξίας των αγορών
9.Enter a Supplier payment	Με τη λειτουργία αυτή εισάγουμε μια νέα πληρωμή σε ένα προμηθευτή η οποία καταχωρείται στο αρχείο paymentsupplier.txt με την εκτέλεση της συνάρτησης insert_paymentsupplier(). Αρχικά εισάγεται και ελέγχεται ο κωδικός του προμηθευτή και μετά εισάγονται η ημερομηνία (στην οποία γίνεται επίσης έλεγχος ορθής καταχώρισης) και το ποσό πληρωμής.
10.Print Supplier's Payments	Με την επιλογή αυτή εμφανίζουμε στην οθόνη τις πληρωμές σε όλους τους προμηθευτές. Η λειτουργία αυτή υλοποιείται με την εκτέλεση της συνάρτησης insert_paymentsupplier(). Για κάθε πληρωμή εμφανίζονται τα στοιχεία του προμηθευτή, το ποσό της και η ημερομηνία που έγινε.
11.Return to main menu	Επιστρέφουμε στο βασικό μενού του προγράμματος που φαίνεται στην Εικόνα 1-Βασικό Μενού Χειρισμού Προγράμματος

13.7.3 Περιγραφή Λειτουργιών του υπομενού Ειδών

Επιλέγοντας 3 στο Βασικό Μενού Χειρισμού του Προγράμματος ο χρήστης έχει τη δυνατότητα να εργαστεί με τους προμηθευτές της επιχείρησης. Συγκεκριμένα με την επιλογή του Items's Menu εμφανίζεται στο χρήστη το ακόλουθο υπομενού:

```
cout<<"ITEMS MENU\n";
cout<<"=================\n";


cout<<"0.Delete all Items\n";
cout<<"1.Insert New Item\n";
cout<<"2.Print Items\n";
cout<<"3.Update Item\n";
```

```
cout<<"4.Delete Item\n";
cout<<"5.Show Rest of a specific Item\n";
cout<<"6.Print Items in Lack\n";
cout<<"7.Print Item Sales\n";
cout<<"8.Print Item Purchases\n";
cout<<"9.Return to main menu\n";


cout<<"Choose (1-9): ";
cin>>ch;
```

Οι λειτουργίες αυτού του υπομενού περιγράφονται στον ακόλουθο πίνακα:

Επιλογή	**Αποτέλεσμα**
0.Delete all Items	Διαγράφεται το αρχείο ειδών με την εκτέλεση της συνάρτησης delete_items(). Στη συνάρτηση αυτή η διαγραφή του αρχείου item.txt γίνονται με την ακόλουθη δήλωση: ofstream acf("items.txt"); που δεν υπάρχουν καταχωρημένοι πλέον στο σύστημα
1.Insert New Item	Με τη λειτουργία αυτή καταχωρείται ένα νέο είδος στο σύστημα. Αυτό γίνεται με την εκτέλεση της συνάρτησης insert_item(). Κάθε νέο είδος καταχωρείται στο αρχείο item.txt.
2.Print Items	Με την επιλογή αυτή εμφανίζονται στην οθόνη όλα τα είδη που εμπορεύεται η επιχείρηση τα οποία όπως έχουμε αναφέρει είναι καταχωρημένα στο αρχείο items.txt. Αυτό γίνεται με την εκτέλεση της συνάρτησης print_items(). Για το κάθε είδος εκτυπώνονται το όνομα του, ο κωδικός του, το κόστος/τεμάχιο και το υπόλοιπο του.
3. Update Item	Με τη λειτουργία αυτή τροποποιείται το κόστος ενός είδους της επιχείρησης με την εκτέλεση της συνάρτησης update_item(itt). Αρχικά ζητείται ο κωδικός του είδους και ελέγχεται αν αυτός καταχωρημένος στο αρχείο ειδών (item.txt). Αν ναι τότε ζητείται από

	το χρήστη να εισάγει μια νέα τιμή κόστους η οποία αντικαθιστά την υπάρχουσα τιμή του είδους.
4.Delete Item	Με την επιλογή αυτή διαγράφεται ένα είδος της επιχείρησης με την εκτέλεση της συνάρτησης delete_item(delitt). Το όρισμα delitt της συνάρτησης είναι ο κωδικός του είδους που θέλουμε να διαγράψουμε. Πρώτα ελέγχεται αν το είδος υπάρχει κάνοντας έλεγχο στον κωδικό του και εφόσον αυτό συμβαίνει τότε το πρόγραμμα το διαγράφει από το αρχείο item.txt ζητώντας πρώτα την επιβεβαίωση του χρήστη.
5.Show Rest of a specific Item	Με τη λειτουργία αυτή τυπώνεται το υπόλοιπο ενός είδους με την εκτέλεση της συνάρτησης print_items_rest(cod) η οποία ανοίγει το αρχείο item.txt, ελέγχει αν το είδος είναι καταχωρημένο σε αυτό και αν ναι τυπώνει το τρέχον υπόλοιπο του στην αποθήκη. Αυτή τη πληροφορία είναι πολύ χρήσιμη για μια επιχείρηση διότι έτσι ενημερώνεται για το απόθεμα των ειδών της.
6.Print Items in Lack	Η λειτουργία αυτή μας επιτρέπει να ενημερωθούμε για τα είδη της εταιρίας που βρίσκονται σε έλλειψη καλώντας τη συνάρτηση print_items_in_lack(int limit). Η παράμετρος limit της συνάρτησης είναι το όριο κάτω από το οποίο το είδος θεωρείται ότι είναι σε έλλειψη. Αρχικά εισάγεται το όριο έλλειψης και στη συνέχεια τυπώνονται όλα τα είδη που το υπόλοιπο τους είναι μικρότερο από το όριο αυτό. Αυτή τη πληροφορία είναι επίσης πολύ χρήσιμη για μια επιχείρηση διότι έτσι ενημερώνεται για τα είδη που έχουν χαμηλό απόθεμα ώστε να τα παραγγείλει από προμηθευτές της
7.Print Item Sales	Η λειτουργία αυτή εμφανίζει όλες τις πωλήσεις που έχει πραγματοποιήσει η επιχείρηση σε ένα συγκεκριμένο είδος με την εκτέλεση της συνάρτησης print_item_sales(icode). Αρχικά ζητείται ο κωδικός του είδους, μετά ελέγχεται η ύπαρξη του κωδικού αυτού και τέλος ανοίγει το αρχείο salecustomer.txt μέσα από το οποίο επιλέγονται και τυπώνονται όλες οι πωλήσεις στο συγκεκριμένο είδος. Για την κάθε πώληση εμφανίζονται τα στοιχεία του πελάτη στον οποίο πραγματοποιήθηκε καθώς επίσης και η αξία της.
8.Print Item Purchases	Η λειτουργία αυτή εμφανίζει όλες τις αγορές που έχει πραγματοποιήσει η επιχείρηση για ένα συγκεκριμένο είδος με την

	εκτέλεση της συνάρτησης print_item_purchases(itemcode). Αρχικά ζητείται ο κωδικός του είδους, μετά ελέγχεται η ύπαρξη του κωδικού αυτού και τέλος ανοίγει το αρχείο salesupplier.txt μέσα από το οποίο επιλέγονται και τυπώνονται όλες οι αγορές στο συγκεκριμένο είδος. Για την κάθε αγορά εμφανίζονται τα στοιχεία του προμηθευτή από τον οποίο πραγματοποιήθηκε καθώς επίσης και η αξία της. Αυτή και η προηγούμενη λειτουργία αποτελούν σημαντικές πληροφορίες για μια επιχείρηση προκειμένου αυτή να έχει μια ακριβή εικόνα για τις κινήσεις των ειδών της.
9.Return to main menu	Επιστρέφουμε στο βασικό μενού του προγράμματος που φαίνεται στην Εικόνα 1-Βασικό Μενού Χειρισμού Προγράμματος

Σημείωση

Η συνάρτηση cls() που υπάρχει και στα 3 προηγούμενα υπομενού χρησιμοποιείται για τον «καθαρισμό» της οθόνης από τα προηγούμενα αποτελέσματα σε κάθε επαναληπτική εκτέλεση του προγράμματος

2.2 Περιγραφή Κλάσεων Πληροφοριακού Συστήματος

Ακολούθως δίνεται ένας πίνακας με όλες τις κλάσεις του προγράμματος και μια σύντομη περιγραφή γιαυτές. Οι κλάσεις αυτές ορίζονται στο header file epixeir.dat

2.2.1 Κλάση ClientData

Κλάση ClientData	**Περιγραφή**
class ClientData { private: int codecustomer; char lastname[30]; char firstname[20]; char adresscode[30]; char telephonenumber[10]; double balance; public: ClientData (int=0, char[]= "", char[]="", char[]= "", char[]= "", double=0.0);	Στην κλάση ClientData ορίζονται τα στοιχεία που περιγράφουν ένα πελάτη της επιχείρησης. **Ιδιωτικά μέλη της κλάσης** · ο κωδικός του πελάτη (codecustomer) · το επώνυμο του πελάτη (lastname[30]) · το όνομα του πελάτη (firstname[20]) · η διεύθυνση του πελάτη (adresscode[30]) · το τηλέφωνο πελάτη (telephonenumber[10]) · το υπόλοιπο του πελάτη στην επιχείρηση (balance) **Δημιουργός με προεπιλεγμένες τιμές στα ορίσματα του για αρχικοποίηση των ιδιωτικών μελών της κλάσης ClientData** Ο δημιουργός καλείται με 2 τρόπους: είτε με ορίσματα οπότε γίνεται συγκεκριμένη αρχικοποίηση στα ιδιωτικά μέλη είτε χωρίς ορίσματα οπότε μπαίνουν κενές τιμές στα μέλη.

	Μέθοδοι set που τροποποιούν τις τιμές των ιδιωτικών μελών της κλάσης ClientData
void set_codecustomer(int);	· Μέθοδος set_codecustomer(int) που τροποποιεί το μέλος codecustomer (κωδικός) ενός πελάτη
void set_lastname(char[]);	· Μέθοδος set_lastname(char[]) που τροποποιεί το μέλος lastname (επώνυμο) ενός πελάτη
void set_firstname(char[]);	· Μέθοδος set_firstname(char[]) που τροποποιεί το μέλος firstname (όνομα) ενός πελάτη
void set_adresscode(char[]);	· Μέθοδος set_adresscode(char[]) που τροποποιεί το μέλος adresscode (διεύθυνση) ενός πελάτη
void set_telephonenumber(char[]);	· Μέθοδος set_telephonenumber(char[]) που τροποποιεί το μέλος telephonenumber (τηλέφωνο) ενός πελάτη
void set_balance(double);	· Μέθοδος set_balance (double) που τροποποιεί το μέλος balance (υπόλοιπο) ενός πελάτη
	Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης ClientData
	· Μέθοδος get_codecustomer() που επιστρέφει τον κωδικό ενός πελάτη
	· Μέθοδος get_lastname() που επιστρέφει το μέλος επώνυμο (lastname) ενός πελάτη

int get_codecustomer();
 	
char* get_lastname();
 	
char* get_firstname();
 	
char* get_adresscode();
 	
char* get_telephonenumber();
 	
double get_balance();
 	
};
	· Μέθοδος get_firstname() που επιστρέφει το όνομα (firstname) ενός πελάτη
	
	· Μέθοδος get_adresscode() που επιστρέφει τη διεύθυνση ενός πελάτη
	
	· Μέθοδος get_telephonenumber() που επιστρέφει το τηλέφωνο ενός πελάτη
	
	· Μέθοδος get_balance() που επιστρέφει το υπόλοιπο ενός πελάτη

2.2.2 Κλάση SupplierData

```
class SupplierData
{

private:
        int codesupplier;

        char lastname[30];

        char firstname[20];

        char adresscode[30];


        char telephonenumber[10];

        double balance;




public:
SupplierData (int=0, char[]="",
char[]="",char[]="", char[]="", double=0.0);
```

Στην κλάση αυτή καταχωρούνται τα στοιχεία που περιγράφουν ένα προμηθευτή της επιχείρησης

Ιδιωτικά μέλη της κλάσης SupplierData

- ο κωδικός του προμηθευτή (codesupplier)
- το επώνυμο του προμηθευτή (lastname[30])
- το όνομα του προμηθευτή (firstname[30])
- η διεύθυνση του προμηθευτή (adresscode[30])
- το τηλέφ. προμηθευτή (telephonenumber[10])
- το υπόλοιπο της επιχείρησης στον προμηθευτή (balance)

Δημιουργός με προεπιλεγμένες τιμές στα ορίσματα του για αρχικοποίηση των ιδιωτικών μελών της κλάσης ClientData

Ο δημιουργός καλείται με 2 τρόπους: είτε με ορίσματα οπότε γίνεται συγκεκριμένη αρχικοποίηση στα ιδιωτικά μέλη είτε χωρίς ορίσματα οπότε μπαίνουν κενές τιμές στα μέλη

Μέθοδοι set που τροποποιούν τις τιμές των ιδιωτικών μελών της κλάσης

- Μέθοδος set_codesupplier(int) που τροποποιεί το μέλος codesupplier (κωδικός) ενός προμηθευτή
- Μέθοδος set_lastname(char[]) που τροποποιεί το

void set_codesupplier(int);
void set_lastname(char[]);
void set_firstname(char[]);
void set_adresscode(char[]);
void set_telephonenumber(char[]);
void set_balance(double);
	μέλος lastname (επώνυμο) ενός προμηθευτή
	· Μέθοδος set_firstname(char[]) που τροποποιεί το μέλος firstname (όνομα) ενός προμηθευτή
	· Μέθοδος set_adresscode(char[]) που τροποποιεί το μέλος adresscode (διεύθυνση) ενός προμηθευτή
	· Μέθοδος set_telephonenumber(char[]) που τροποποιεί το μέλος telephonenumber (τηλέφωνο) ενός προμηθευτή
	· Μέθοδος set_balance (double) που τροποποιεί το μέλος balance (υπόλοιπο) σε ένα προμηθευτή
	Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης SupplierData
	· Μέθοδος get_codecustomer() που επιστρέφει τον κωδικό ενός προμηθευτή
	· Μέθοδος get_lastname() που επιστρέφει το μέλος επώνυμο (lastname) ενός προμηθευτή
	· Μέθοδος get_firstname() που επιστρέφει το όνομα (firstname) ενός προμηθευτή
	· Μέθοδος get_adresscode() που επιστρέφει τη διεύθυνση ενός προμηθευτή
	· Μέθοδος get_telephonenumber() που επιστρέφει το τηλέφωνο ενός προμηθευτή

int get_codesupplier();
 	
char* get_lastname();
 	
char* get_firstname();
 	
char* get_adresscode();
 	
char* get_telephonenumber();
 	
double get_balance();
};
	· Μέθοδος get_balance() που επιστρέφει το υπόλοιπο ενός προμηθευτή

2.2.3 Κλάση ItemData

class ItemData {	Στην κλάση αυτή καταχωρούνται τα στοιχεία που περιγράφουν ένα είδος της επιχείρησης. **<u>Ιδιωτικά μέλη της κλάσης ItemData</u>**

private:
int itemcode;
char itemname[30];
double cost;
int rest;
public:
ItemData(int=0, char[]="", double=0.0, int=0);
void set_itemcode(int);
void set_itemname(char[]);
	· ο κωδικός του είδους (itemcode)
	· η ονομασία του είδους (itemname[30])
	· το κόστος του είδους
	· το υπόλοιπο του είδους
	Δημιουργός με προεπιλεγμένες τιμές στα ορίσματα του για αρχικοποίηση των ιδιωτικών μελών της κλάσης ItemData
	Ο δημιουργός καλείται με 2 τρόπους: είτε με ορίσματα οπότε γίνεται συγκεκριμένη αρχικοποίηση στα ιδιωτικά μέλη είτε χωρίς ορίσματα οπότε μπαίνουν κενές τιμές στα μέλη
	Μέθοδοι set που τροποποιούν τις τιμές των ιδιωτικών μελών της κλάσης
	· Μέθοδος set_item(int) που τροποποιεί το μέλος itemcode (κωδικός) ενός είδους
	· Μέθοδος set_itemname(char[]) που τροποποιεί το μέλος itemname (ονομασία) ενός είδους
	· Μέθοδος set_cost(double) που τροποποιεί το μέλος cost (κόστος) ενός είδους
	· Μέθοδος set_rest(double) που τροποποιεί το μέλος rest (υπόλοιπο) ενός είδους

void set_cost(double);
 	
void set_rest(int);
 	
int get_itemcode();
 	
char* get_itemname();
 	
double get_cost();
 	
int get_rest();
 	
};
	<u>Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης ItemData</u>
	
	· Μέθοδος get_codecustomer() που επιστρέφει τον κωδικό ενός είδους
	
	· Μέθοδος get_codecustomer() που επιστρέφει την ονομασία ενός είδους
	
	· Μέθοδος get_cost() που επιστρέφει το κόστος ενός είδους
	
	· Μέθοδος get_rest() που επιστρέφει το υπόλοιπο ενός είδους

2.2.4 Κλάση PaymentCustomerData

class PaymentCustomerData { private: int codecustomer; char lastname[30]; char firstname[20]; double amount; Date date; public: PaymentCustomerData(int=0,char[]="	Στην κλάση PaymentCustomerData ορίζονται τα στοιχεία που περιγράφουν μια πληρωμή ενός πελάτη. **Ιδιωτικά μέλη της κλάσης** · ο κωδικός του πελάτη (codecustomer). Το πεδίο αυτό χρησιμοποιείται για τη συσχέτιση μιας πληρωμής με ένα πελάτη. · το επώνυμο του πελάτη (lastname[30]). Το πεδίο αυτό χρησιμοποιείται για να εμφανίζει το επώνυμο του πελάτη που κάνει την πληρωμή · το όνομα του πελάτη (firstname[20]). Το πεδίο αυτό χρησιμοποιείται για να εμφανίζει το όνομα του πελάτη που κάνει την πληρωμή · το ποσό πληρωμής (amount) · η ημερομηνία πληρωμής (date) **Δημιουργός με προεπιλεγμένες τιμές στα ορίσματα του για αρχικοποίηση των ιδιωτικών μελών της κλάσης PaymentCustomerData** Ο δημιουργός καλείται με 2 τρόπους: είτε με ορίσματα οπότε γίνεται συγκεκριμένη αρχικοποίηση στα ιδιωτικά μέλη είτε χωρίς ορίσματα οπότε μπαίνουν κενές τιμές στα μέλη **Μέθοδοι set που τροποποιούν τις τιμές των ιδιωτικών μελών της κλάσης PaymentCustomerData** · Μέθοδος set_codecustomer(int) που τροποποιεί το

",char[]="",double=0.0,int=0,int=0,int=0);	μέλος codecustomer (κωδικός) του πελάτη που κάνει την πληρωμή
void set_codecustomer(int);	· Μέθοδος set_lastname(char[]) που τροποποιεί το μέλος lastname (επώνυμο) του πελάτη που κάνει την πληρωμή · Μέθοδος set_firstname(char[]) που τροποποιεί το μέλος firstname (όνομα) του πελάτη που κάνει την πληρωμή
void set_lastname(char[]);	· Μέθοδος set_amount(double) που τροποποιεί το μέλος amount, δηλαδή αλλάζει το ποσό μιας πληρωμής · Μέθοδος set_date(int, int ,int) που τροποποιεί το μέλος date, δηλαδή αλλάζει την ημερομηνία μιας πληρωμής
void set_firstname(char[]);	**Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης PaymentCustomerData**
void set_amount(double);	· Μέθοδος get_codecustomer() που επιστρέφει τον κωδικό του πελάτη που κάνει την πληρωμή · Μέθοδος get_lastname() που επιστρέφει το επώνυμο του πελάτη που κάνει την πληρωμή
void set_date(int, int, int);	· Μέθοδος get_lastname() που επιστρέφει το όνομα του πελάτη που κάνει την πληρωμή

int get_codecustomer();
 	
char* get_lastname();
 	
char* get_firstname();
 	
double get_amount();
 	
int get_day();
 	
int get_month();
 	
int get_year();
	· Μέθοδος get_amount() που επιστρέφει το ποσό πληρωμής ενός πελάτη
	
	· Μέθοδος get_day() που επιστρέφει την ημέρα πληρωμής ενός πελάτη
	
	· Μέθοδος get_month() που επιστρέφει το μήνα πληρωμής ενός πελάτη
	
	· Μέθοδος get_year() που επιστρέφει το έτος πληρωμής ενός πελάτη

};	

2.2.5 Κλάση PaymentSupplierData

class PaymentSuppplierData { private: int codesupplier; char lastname[30]; char firstname[20]; double amount;	Στην κλάση PaymentSupplierData ορίζονται τα στοιχεία που περιγράφουν μια πληρωμή της εταιρίας σε ένα προμηθευτή της. **Ιδιωτικά μέλη της κλάσης** · ο κωδικός του προμηθευτή (codesupplier). Το πεδίο αυτό χρησιμοποιείται για τη συσχέτιση μιας πληρωμής με ένα προμηθευτή. · το επώνυμο του προμηθευτή (lastname[30]). Το πεδίο αυτό χρησιμοποιείται για να εμφανίζει το επώνυμο του προμηθευτή στον οποίο γίνεται η πληρωμή · το όνομα του προμηθευτή (firstname[20]). Το πεδίο αυτό χρησιμοποιείται για να εμφανίζει το όνομα του προμηθευτή στον οποίο γίνεται η πληρωμή · το ποσό πληρωμής (amount) · η ημερομηνία πληρωμής (date) **Δημιουργός με προεπιλεγμένες τιμές στα ορίσματα του για αρχικοποίηση των ιδιωτικών μελών της κλάσης PaymentSupplierData** Ο δημιουργός καλείται με 2 τρόπους: είτε με ορίσματα

Date date;	οπότε γίνεται συγκεκριμένη αρχικοποίηση στα ιδιωτικά μέλη είτε χωρίς ορίσματα οπότε μπαίνουν κενές τιμές στα μέλη

public:
PaymentSupplierData(int=0,char[]="",char[]="",double=0.0,int=0,int=0,int=0);
void set_codesupplier(int);
void set_lastname(char[]);
void set_firstname(char[]);
	<u>Μέθοδοι set που τροποποιούν τις τιμές των ιδιωτικών μελών της κλάσης PaymentSupplierData</u>
	· Μέθοδος set_codesupplier(int) που τροποποιεί το μέλος codecustomer (κωδικός) του προμηθευτή το επώνυμο του προμηθευτή στον οποίο γίνεται η πληρωμή
	· Μέθοδος set_lastname(char[]) που τροποποιεί το μέλος lastname (επώνυμο) το επώνυμο του προμηθευτή στον οποίο γίνεται η πληρωμή
	· Μέθοδος set_firstname(char[]) που τροποποιεί το μέλος firstname (όνομα) το επώνυμο του προμηθευτή στον οποίο γίνεται η πληρωμή
	· Μέθοδος set_amount(double) που τροποποιεί το μέλος amount, δηλαδή αλλάζει το ποσό μιας πληρωμής
	· Μέθοδος set_date(int, int ,int) που τροποποιεί το μέλος date, δηλαδή αλλάζει την ημερομηνία μιας πληρωμής
	<u>Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης PaymentCustomerData</u>
	· Μέθοδος get_codecustomer() που επιστρέφει τον

void set_amount(double);
 	
void set_date(int, int, int);
 	
int get_codecustomer();
 	
char* get_lastname();
 	
char* get_firstname();
 	
double get_amount();
	κωδικό του το επώνυμο του προμηθευτή στον οποίο γίνεται η πληρωμή
	
	· Μέθοδος get_lastname() που επιστρέφει το επώνυμο του προμηθευτή στον οποίο γίνεται η πληρωμή
	
	· Μέθοδος get_lastname() που επιστρέφει το όνομα του προμηθευτή στον οποίο γίνεται η πληρωμή
	
	· Μέθοδος get_amount() που επιστρέφει το ποσό πληρωμής του προμηθευτή στον οποίο γίνεται η πληρωμή
	
	· Μέθοδος get_day() που επιστρέφει την ημέρα πληρωμής σε ένα προμηθευτή
	
	· Μέθοδος get_month() που επιστρέφει το μήνα πληρωμής σε ένα προμηθευτή
	
	· Μέθοδος get_year() που επιστρέφει το έτος πληρωμής σε ένα προμηθευτή

int get_day();
 	
int get_month();
 	
int get_year();
 	
};	

2.2.6 Κλάση SaleCustomerData

class SaleCustomerData { private: int codecustomer; char lastname[30];	Στην κλάση αυτή καταχωρούνται τα στοιχεία που αφορούν μια πώληση σε πελάτη. **<u>Ιδιωτικά μέλη της κλάσης</u>** · ο κωδικός του πελάτη (codecustomer). Το πεδίο αυτό χρησιμοποιείται για τη συσχέτιση μιας πώλησης με ένα πελάτη. · το επώνυμο του πελάτη (lastname[30]). Το πεδίο αυτό χρησιμοποιείται για να εμφανίζει το επώνυμο του πελάτη στον οποίο γίνεται η πώληση · το όνομα του πελάτη (lastname[30]). Το πεδίο αυτό χρησιμοποιείται για να εμφανίζει το όνομα του πελάτη στον οποίο γίνεται η πώληση

char firstname[20]; double amount; Date date; int itemcode; int quantity; public: SaleCustomerData(int=0, char[]="", char[]="", double=0.0, int=0, int=0, int=0, int=0, int=0);	· η αξία της πώλησης (amount) · η ημερομηνία πώλησης (date) · ο κωδικός του είδους (itemcode) το οποίο πωλείται. Το πεδίο αυτό χρησιμοποιείται για τη συσχέτιση μιας πώλησης με ένα είδος · η ποσότητα του είδους (itemcode) το οποίο πωλείται. **Δημιουργός με προεπιλεγμένες τιμές στα ορίσματα του για αρχικοποίηση των ιδιωτικών μελών της κλάσης SaleCustomerData** Ο δημιουργός καλείται με 2 τρόπους: είτε με ορίσματα οπότε γίνεται συγκεκριμένη αρχικοποίηση στα ιδιωτικά μέλη είτε χωρίς ορίσματα οπότε μπαίνουν κενές τιμές στα μέλη **Μέθοδοι set που τροποποιούν τις τιμές των ιδιωτικών μελών της κλάσης SaleCustomerData** · Μέθοδος set_codecustomer(int) που τροποποιεί το μέλος codecustomer (κωδικός) του πελάτη στον οποίο γίνεται η πώληση · Μέθοδος set_lastname(char []) που τροποποιεί το μέλος lastname (επώνυμο) του πελάτη στον οποίο γίνεται η πώληση · Μέθοδος set_firstname(char []) που τροποποιεί το μέλος firstname (όνομα) του πελάτη στον οποίο

	γίνεται η πώληση
void set_codecustomer(int);	· Μέθοδος set_amount(double) που τροποποιεί το μέλος amount, δηλαδή αλλάζει της αξία μιας πώλησης
void set_lastname(char[]);	· Μέθοδος set_date(int, int ,int) που τροποποιεί το μέλος date, δηλαδή αλλάζει την ημερομηνία μιας πώλησης
void set_firstname(char[]);	· Μέθοδος set_itemcode(int) που τροποποιεί το μέλος itemcode (κωδικός) του είδους το οποίο πωλείται · Μέθοδος set_quantity(int) που τροποποιεί το μέλος quantity (ποσότητα) του είδους το οποίο πωλείται
void set_amount(double);	**Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης SaleCustomerData**
void set_date(int, int, int);	· Μέθοδος get_codecustomer() που επιστρέφει τον κωδικό του πελάτη στον οποίο γίνεται η πώληση · Μέθοδος get_lastname() που επιστρέφει το επώνυμο του πελάτη στον οποίο γίνεται η πώληση
void set_itemcode(int);	· Μέθοδος get_firstname() που επιστρέφει το όνομα του πελάτη στον οποίο γίνεται η πώληση · Μέθοδος get_amount() που επιστρέφει την αξία μιας πώλησης
	· Μέθοδος get_itemcode() που επιστρέφει τον κωδικό του είδους που πωλείται

void set_quantity(int);
 	
int get_codecustomer();
 	
char* get_lastname();
 	
char* get_firstname();
 	
double get_amount();
 	
int get_itemcode();
 	
int get_quantity();
 	
int get_day();
	· Μέθοδος get_quantity() που επιστρέφει την ποσότητα του είδους που πωλείται
	
	· Μέθοδος get_day() που επιστρέφει την ημέρα της πώλησης
	
	· Μέθοδος get_month() που επιστρέφει το μήνα της πώλησης
	
	· Μέθοδος get_year() που επιστρέφει το έτος της πώλησης

int get_month();
 	
int get_year();
 	
};	

2.2.7 Κλάση SaleSupplierData

class SaleSupplierData { private: int codesupplier; char lastname[30]; char firstname[20];	Στην κλάση αυτή καταχωρούνται τα στοιχεία που αφορούν μια αγορά από προμηθευτή. **Ιδιωτικά μέλη της κλάσης** · ο κωδικός του προμηθευτή (codesupplier). Το πεδίο αυτό χρησιμοποιείται για τη συσχέτιση μιας αγοράς με ένα προμηθευτή. · το επώνυμο του προμηθευτή (lastname[30]). Το πεδίο αυτό χρησιμοποιείται για να εμφανίζει το επώνυμο του προμηθευτή από τον οποίο γίνεται η αγορά · το όνομα του προμηθευτή (lastname[30]). Το πεδίο αυτό χρησιμοποιείται για να εμφανίζει το όνομα του προμηθευτή από τον οποίο γίνεται η αγορά · η αξία της αγοράς (amount)

double amount; Date date; int itemcode; int quantity; public: SaleCustomerData(int=0, char[]="", char[]="", double=0.0, int=0, int=0, int=0, int=0, int=0);	· η ημερομηνία αγοράς (date) · ο κωδικός του είδους (itemcode) το οποίο αγοράζεται. Το πεδίο αυτό χρησιμοποιείται για τη συσχέτιση μιας αγοράς με ένα είδος · η ποσότητα του είδους (itemcode) το οποίο αγοράζεται. **Δημιουργός με προεπιλεγμένες τιμές στα ορίσματα του για αρχικοποίηση των ιδιωτικών μελών της κλάσης SaleSupplierData** Ο δημιουργός καλείται με 2 τρόπους: είτε με ορίσματα οπότε γίνεται συγκεκριμένη αρχικοποίηση στα ιδιωτικά μέλη είτε χωρίς ορίσματα οπότε μπαίνουν κενές τιμές στα μέλη **Μέθοδοι set που τροποποιούν τις τιμές των ιδιωτικών μελών της κλάσης SaleSupplierData** · Μέθοδος set_codesupplier(int) που τροποποιεί το μέλος codesupplier (κωδικός) του προμηθευτή από τον οποίο γίνεται η αγορά · Μέθοδος set_lastname(char []) που τροποποιεί το μέλος lastname (επώνυμο) του προμηθευτή από τον οποίο γίνεται η αγορά · Μέθοδος set_firstname(char []) που τροποποιεί το μέλος firstname (όνομα) του προμηθευτή από τον οποίο γίνεται η αγορά

void set_codecustomer(int);
 	
void set_lastname(char[]);
 	
void set_firstname(char[]);
 	
void set_amount(double);
 	
void set_date(int, int, int);
 	
void set_itemcode(int);
 	
void set_quantity(int);
	· Μέθοδος set_amount(double) που τροποποιεί το μέλος amount, δηλαδή αλλάζει της αξία μιας αγοράς
	
	· Μέθοδος set_date(int, int ,int) που τροποποιεί το μέλος date, δηλαδή αλλάζει την ημερομηνία μιας αγοράς
	
	· Μέθοδος set_itemcode(int) που τροποποιεί το μέλος itemcode (κωδικός) του είδους το οποίο αγοράζεται
	
	· Μέθοδος set_quantity(int) που τροποποιεί το μέλος quantity (ποσότητα) του είδους το οποίο αγοράζεται
	
	Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης SaleSupplierData
	
	· Μέθοδος get_codesupplier() που επιστρέφει τον κωδικό του προμηθευτή από τον οποίο γίνεται η αγορά
	
	· Μέθοδος get_lastname() που επιστρέφει το επώνυμο του προμηθευτή από τον οποίο γίνεται η αγορά
	
	· Μέθοδος get_firstname() που επιστρέφει το όνομα του προμηθευτή από τον οποίο γίνεται η αγορά
	
	· Μέθοδος get_amount() που επιστρέφει την αξία μιας αγοράς
	
	· Μέθοδος get_itemcode() που επιστρέφει τον κωδικό

	του είδους που αγοράζεται · Μέθοδος get_quantity() που επιστρέφει την ποσότητα του είδους που αγοράζεται · Μέθοδος get_day() που επιστρέφει την ημέρα της αγοράς · Μέθοδος get_month() που επιστρέφει το μήνα της αγοράς Μέθοδος get_year() που επιστρέφει το έτος της αγοράς

int get_codesupplier();
 	
char* get_lastname();
 	
char* get_firstname();
 	
double get_amount();
 	
int get_itemcode();
 	
int get_quantity();	

int get_day();
 	
int get_month();
 	
int get_year();
 	
};	

2.2.8 Κλάση Date

Η κλάση Date ορίζεται στο header file Date.h και περιγράφει μια ημερομηνία ως μια συλλογή από 3 πεδία: day (ημέρα), month (έτος) και year (έτος) σύμφωνα με την ακόλουθη δήλωση:

```
class Date
{
public:
    int day;
    int month;
    int year;
};
```

2.2.9 Κλάση Transaction

Η κλάση Transaction ορίζεται στο header file Transaction.h και περιγράφει μια συναλλαγή (είσπραξη ή πληρωμή) ως μια συλλογή από 4 πεδία: code (κωδικός πελάτη ή προμηθευτή που εκτελεί την κίνηση), amount (ποσό είσπραξης ή πληρωμής), type (τύπος κίνησης) και date (ημερομηνία κίνησης) σύμφωνα με την ακόλουθη δήλωση:

```
class Transaction
{
    int code;
    double amount;
    char type[20];
    char date[10];
};
```

2.3 Περιγραφή-Υλοποίηση Συναρτήσεων Πληροφοριακού Συστήματος

2.3.1 Αρχείο clients.cpp με υλοποίηση των μεθόδων κλάσης ClientData

Όλες οι μέθοδοι που δηλώθηκαν στην κλάση ClientData (στο αρχείο epixeir.h) υλοποιούνται στο αρχείο clients.cpp και εξηγούνται στον ακόλουθο πίνακα

ClientData::ClientData(int c, char l[], char f[], char a[], char t[], double b)
{
codecustomer=c;
strcpy(lastname,l);
strcpy(firstname,f);
strcpy(adresscode, a);
strcpy(telephonenumber, t);
balance=b;
}
	· Δημιουργός της κλάσης ClientData που θέτει τα ορίσματα (τυπικές παραμέτρους) που λαμβάνει στα μέλη της κλάσης

void ClientData::set_codecustomer(int c)
{
codecustomer=c;
}
	· Υλοποίηση μεθόδου που τροποποιεί τον κωδικό ενός πελάτη

void ClientData::set_lastname(char l[])
{
strcpy(lastname,l);
}
	· Υλοποίηση μεθόδου που τροποποιεί το επώνυμο ενός πελάτη

void ClientData::set_firstname(char f[])
{
strcpy(firstname,f);
}
	· Υλοποίηση μεθόδου που τροποποιεί το όνομα ενός πελάτη

void ClientData::set_adresscode(char a[])	

{
strcpy(adresscode, a);
}
 	
void ClientData::set_telephonenumber(char t[])
{
strcpy(telephonenumber, t);
}
 	
void ClientData::set_balance(double b)
{
balance=b;
}
 	
int ClientData::get_codecustomer()
{
return codecustomer;
}
 	
char* ClientData::get_lastname()
{
return lastname;
}
 	
char* ClientData::get_firstname()
{
return firstname;
}
 	
char* ClientData::get_adresscode()
{
return adresscode;
}
	· Υλοποίηση μεθόδου που τροποποιεί τη διεύθυνση ενός πελάτη
	
	· Υλοποίηση μεθόδου που τροποποιεί το τηλέφωνο ενός πελάτη
	
	· Υλοποίηση μεθόδου που τροποποιεί το υπόλοιπο ενός πελάτη
	
	· Υλοποίηση μεθόδου που επιστρέφει το κωδικό ενός πελάτη
	
	· Υλοποίηση μεθόδου που επιστρέφει το επώνυμο ενός πελάτη
	
	· Υλοποίηση μεθόδου που επιστρέφει το όνομα ενός πελάτη

char* ClientData::get_telephonenumber()
{
return telephonenumber;
}
 	
double ClientData::get_balance()
{
return balance;
}
	· Υλοποίηση μεθόδου που επιστρέφει τη διεύθυνση ενός πελάτη
	
	· Υλοποίηση μεθόδου που επιστρέφει τη διεύθυνση ενός πελάτη
	
	· Υλοποίηση μεθόδου που επιστρέφει το υπόλοιπο ενός πελάτη

2.3.2 Αρχείο suppliers.cpp με υλοποίηση των μεθόδων κλάσης SupplierData

Όλες οι μέθοδοι που δηλώθηκαν στην κλάση SupplierData (στο αρχείο epixeir.h) υλοποιούνται στο αρχείο suppliers.cpp και εξηγούνται στον ακόλουθο πίνακα

SupplierData::SupplierData(int c, char l[], char f[], char a[], char t[], double b)
{
codecustomer=c;
strcpy(lastname,l);
strcpy(firstname,f);
strcpy(adresscode, a);
strcpy(telephonenumber, t);
	· Δημιουργός της κλάσης SupplierData που θέτει τα ορίσματα (τυπικές παραμέτρους) που λαμβάνει στα μέλη της κλάσης

balance=b;
}
 	
void SupplierData::set_codecustomer(int c)
{
codecustomer=c;
}
 	
void SupplierData::set_lastname(char l[])
{
strcpy(lastname,l);
}
 	
void SupplierData::set_firstname(char f[])
{
strcpy(firstname,f);
}
 	
void SupplierData::set_adresscode(char a[])
{
strcpy(adresscode, a);
}
 	
void SupplierData::set_telephonenumber(char t[])
{
strcpy(telephonenumber, t);
}
 	
void SupplierData::set_balance(double b)
{
balance=b;
	· Υλοποίηση μεθόδου που τροποποιεί τον κωδικό ενός προμηθευτή
	· Υλοποίηση μεθόδου που τροποποιεί το επώνυμο ενός προμηθευτή
	· Υλοποίηση μεθόδου που τροποποιεί το όνομα ενός προμηθευτή
	· Υλοποίηση μεθόδου που τροποποιεί τη διεύθυνση ενός προμηθευτή
	· Υλοποίηση μεθόδου που τροποποιεί το τηλέφωνο ενός προμηθευτή
	· Υλοποίηση μεθόδου που τροποποιεί το υπόλοιπο ενός προμηθευτή

}
 	
int SupplierData::get_codecustomer()
{
return codecustomer;
}
	· Υλοποίηση μεθόδου που επιστρέφει το κωδικό ενός προμηθευτή

char* SupplierData::get_lastname()
{
return lastname;
}
	· Υλοποίηση μεθόδου που επιστρέφει το επώνυμο ενός προμηθευτή

char* SupplierData::get_firstname()
{
return firstname;
}
	· Υλοποίηση μεθόδου που επιστρέφει το όνομα ενός προμηθευτή

char* SupplierData::get_adresscode()
{
return adresscode;
}
	· Υλοποίηση μεθόδου που επιστρέφει τη διεύθυνση ενός προμηθευτή

char* SupplierData::get_telephonenumber()
{
return telephonenumber;
}
 	
double SupplierData::get_balance()
{
return balance;
}
	· Υλοποίηση μεθόδου που επιστρέφει τη διεύθυνση ενός προμηθευτή
	
	· Υλοποίηση μεθόδου που επιστρέφει το υπόλοιπο σε ένα προμηθευτή

2.3.3 Αρχείο items.cpp με υλοποίηση των μεθόδων κλάσης ItemData

Όλες οι μέθοδοι που δηλώθηκαν στην κλάση ItemData (στο αρχείο epixeir.h) υλοποιούνται outline στο αρχείο items.cpp και εξηγούνται στον ακόλουθο πίνακα

ItemData::ItemData(int c, char n[], double k,int r)
{
itemcode=c;
strcpy(itemname,n);
cost=k;
rest=r;
}
	· Δημιουργός της κλάσης ItemData που θέτει τα ορίσματα (τυπικές παραμέτρους) που λαμβάνει στα μέλη της κλάσης

void ItemData::set_itemcode(int c)
{
itemcode=c;
}
	· Υλοποίηση μεθόδου που τροποποιεί τον κωδικό ενός είδους

void ItemData::set_itemname(char n[])
{
strcpy(itemname,n);
}
	· Υλοποίηση μεθόδου που τροποποιεί την ονομασία ενός είδους

void ItemData::set_cost(double k)
{
cost=k;
}
	· Υλοποίηση μεθόδου που τροποποιεί το κόστος ενός είδους

void ItemData::set_rest(int r)
{
rest=r;
}
	· Υλοποίηση μεθόδου που τροποποιεί το υπόλοιπο

int ItemData::get_itemcode()
{
return itemcode;
}
 	
char* ItemData::get_itemname()
{
return itemname;
}
 	
double ItemData::get_cost()
{
return cost;
}
 	
int ItemData::get_rest()
{
return rest;
}
	ενός είδους στην αποθήκη της επιχείρησης
	
	· Υλοποίηση μεθόδου που επιστρέφει τον κωδικό ενός είδους
	
	· Υλοποίηση μεθόδου που επιστρέφει την ονομασία ενός είδους
	
	· Υλοποίηση μεθόδου που επιστρέφει το κόστος ενός είδους
	
	· Υλοποίηση μεθόδου που επιστρέφει το υπόλοιπο ενός είδους

2.3.4 Αρχείο salecustomer.cpp με υλοποίηση των μεθόδων κλάσης SaleCustomerData

SaleCustomerData::SaleCustomerData(int cn, char l[],char f[],double am, int day, int month, int year,int itcode, int itqua) {	· Δημιουργός της κλάσης SaleCustomerData που θέτει τα ορίσματα (τυπικές παραμέτρους) που λαμβάνει στα μέλη της κλάσης

codecustomer=cn;
strcpy(lastname,l);
strcpy(firstname,f);
amount=am;
date.day=day;
date.month=month;
date.year=year;
itemcode=itcode;
quantity=itqua;
}	

void SaleCustomerData:: set_codecustomer(int cn)
{
codecustomer=cn;
}
	· Υλοποίηση μεθόδου που τροποποιεί τον κωδικό ενός πελάτη

void SaleCustomerData::set_lastname(char l[])
{
strcpy(lastname,l);
}
	· Υλοποίηση μεθόδου που τροποποιεί το επώνυμο ενός πελάτη

void SaleCustomerData::set_first-name(char f[])
{
strcpy(firstname,f);
}
	· Υλοποίηση μεθόδου που τροποποιεί το όνομα ενός πελάτη

void SaleCustomerData::set_amount(double am)
{
amount=am;
}
	· Υλοποίηση μεθόδου που τροποποιεί την αξία πώλησης σε ένα πελάτη

void SaleCustomerData::set_date(int day, int month, int year)
{
date.day=day;
 	
date.month=month;
 	
date.year=year;
}
	· Υλοποίηση μεθόδου που τροποποιεί την ημερομηνία πώλησης σε ένα πελάτη

void SaleCustomerData::set_itemcode(int itcode)
{
itemcode=itcode;
}
	· Υλοποίηση μεθόδου που τροποποιεί το υπόλοιπο ενός πελάτη

void SaleCustomerData::set_quantity(int itqua)
{
quantity=itqua;
}
	· Υλοποίηση μεθόδου που τροποποιεί την ποσότητα πώλησης σε ένα πελάτη

int SaleCustomerData::get_codecustomer()
{
return codecustomer;
}
	· Υλοποίηση μεθόδου που επιστρέφει τον κωδικό ενός πελάτη

char* SaleCustomerData::get_lastname()
{
return lastname;
}
	· Υλοποίηση μεθόδου που επιστρέφει το επώνυμο ενός πελάτη

char* SaleCustomerData::get_firstname() {	· Υλοποίηση μεθόδου που

return firstname; }	επιστρέφει το όνομα ενός πελάτη

double SaleCustomerData::get_amount()
{
return amount;
}
	· Υλοποίηση μεθόδου που επιστρέφει την αξία μιας πώλησης ενός πελάτη

int SaleCustomerData::get_day()
{
return date.day;
}
	· Υλοποίηση μεθόδου που επιστρέφει την ημέρα που έγινε η πώληση σε ένα πελάτη

int SaleCustomerData::get_month()
{
return date.month;
}
	· Υλοποίηση μεθόδου που επιστρέφει το μήνα που έγινε η πώληση σε ένα πελάτη

int SaleCustomerData::get_year()
{
return date.year;
}
	· Υλοποίηση μεθόδου που επιστρέφει το έτος πώλησης που έγινε η πώληση σε ένα πελάτη

int SaleCustomerData::get_itemcode()
{
return itemcode;
}
	· Υλοποίηση μεθόδου που επιστρέφει τον κωδικό ενός είδους σε μια πώληση

int SaleCustomerData::get_quantity()	· Υλοποίηση μεθόδου που επιστρέφει την ποσότητα μιας πώλησης σε ένα πελάτη

{
return quantity;
}	

2.3.5 Αρχείο salesupplier.cpp με υλοποίηση των μεθόδων κλάσης SaleSuppplierData

SaleSupplierData::SaleSupplierData(int cn, char l[],char f[],double am, int day, int month, int year,int itcode, int itqua)
{
codesupplier=cn;
strcpy(lastname,l);
strcpy(firstname,f);
amount=am;
date.day=day;
date.month=month;
date.year=year;
itemcode=itcode;
quantity=itqua;
 	
}
	· Δημιουργός της κλάσης SaleSupplierData που θέτει τα ορίσματα (τυπικές παραμέτρους) που λαμβάνει στα μέλη της κλάσης

void SaleSupplierData::set_code-supplier (int cn)
{
codesupplier=cn;
}
	· Υλοποίηση μεθόδου που τροποποιεί τον κωδικό ενός προμηθευτή

void SaleSupplierData::set_lastname (char l[])
{
strcpy(lastname,l);
}
	· Υλοποίηση μεθόδου που τροποποιεί το επώνυμο ενός προμηθευτή
	· Υλοποίηση μεθόδου που τροποποιεί το όνομα ενός προμηθευτή

```
void SaleSupplierData::set_firstname (char f[])
{
        strcpy(firstname,f);
}
```

· Υλοποίηση μεθόδου που τροποποιεί την αξία αγοράς από ένα προμηθευτή

```
void SaleSupplierData::set_amount (double am)
{
        amount=am;
}
```

· Υλοποίηση μεθόδου που τροποποιεί την ημερομηνία αγοράς από ένα προμηθευτή

```
void SaleSupplierData::set_date(int day, int month, int year)
{
        date.day=day;

        date.month=month;

        date.year=year;
}
```

· Υλοποίηση μεθόδου που τροποποιεί το υπόλοιπο σε ένα προμηθευτή

```
void SaleSupplierData::set_itemcode(int itcode)
{
        itemcode=itcode;
}
```

· Υλοποίηση μεθόδου που τροποποιεί την ποσότητα αγοράς από ένα προμηθευτή

```
void SaleSupplierData::set_quantity(int itqua)
{
        quantity=itqua;
}
```

· Υλοποίηση μεθόδου που επιστρέφει τον κωδικό ενός προμηθευτή

int SaleSupplierData::get_codesupplier()
{
return codesupplier;
}
	· Υλοποίηση μεθόδου που επιστρέφει το επώνυμο ενός προμηθευτή

char* SaleSupplierData::get_lastname()
{
return lastname;
}
	· Υλοποίηση μεθόδου που επιστρέφει το όνομα ενός προμηθευτή

char* SaleSupplierData::get_firstname()
{
return firstname;
}
	· Υλοποίηση μεθόδου που επιστρέφει την αξία μιας αγοράς από ένα προμηθευτή

double SaleSupplierData::get_amount()
{
return amount;
}
	· Υλοποίηση μεθόδου που επιστρέφει την ημέρα αγοράς από ένα προμηθευτή

int SaleSupplierData::get_day()
{
return date.day;
}
	· Υλοποίηση μεθόδου που επιστρέφει το μήνα αγοράς από ένα προμηθευτή

int SaleSupplierData::get_month()
{
return date.month;
}
	· Υλοποίηση μεθόδου που επιστρέφει το έτος πώλησης αγοράς από ένα προμηθευτή
	· Υλοποίηση μεθόδου που επιστρέφει τον κωδικό ενός είδους σε μια αγορά

int SaleSupplierData::get_year() {	· Υλοποίηση μεθόδου που επιστρέφει την ποσότητα μιας αγοράς από ένα

return date.year;
}
 	
int SaleSupplierData::get_itemcode()
{
return itemcode;
}
 	
int SaleSupplierData::get_quantity()
{
return quantity;
}
	προμηθευτή

2.3.6 Αρχείο paymentcustomer.cpp με υλοποίηση των μεθόδων κλάσης PaymentcustomerData

PaymentCustomerData::PaymentCustomerData(int cn, char l[],char f[],double am, int day, int month, int year)
{
codecustomer=cn;
strcpy(lastname,l);
strcpy(firstname,f);
amount=am;
date.day=day;
date.month=month;
date.year=year;
}
 	
void PaymentCustomerData::set_codecustomer
	· Δημιουργός της κλάσης PaymentCustomerData που θέτει τα ορίσματα (τυπικές παραμέτρους) που λαμβάνει στα μέλη της κλάσης
	
	· Υλοποίηση μεθόδου που τροποποιεί τον κωδικό ενός πελάτη σε μια πληρωμή

(int cn)
{
codecustomer=cn;
}
 	
void PaymentCustomerData::set_lastname (char l[])
{
strcpy(lastname,l);
}
 	
void PaymentCustomerData::set_firstname (char f[])
{
strcpy(firstname,f);
}
 	
void PaymentCustomerData::set_amount (double am)
{
amount=am;
}
 	
void PaymentCustomerData::set_date (int day, int month, int year)
{
date.day=day;
 	
date.month=month;
 	
date.year=year;
}
	· Υλοποίηση μεθόδου που τροποποιεί το επώνυμο ενός πελάτη σε μια πληρωμή
	
	· Υλοποίηση μεθόδου που τροποποιεί το όνομα ενός πελάτη σε μια πληρωμή
	
	· Υλοποίηση μεθόδου που τροποποιεί την αξία μιας πληρωμής από πελάτη
	
	· Υλοποίηση μεθόδου που τροποποιεί την ημερομηνία πληρωμής από ένα πελάτη
	
	· Υλοποίηση μεθόδου που επιστρέφει τον κωδικό ενός πελάτη σε μια πληρωμή
	
	· Υλοποίηση μεθόδου που επιστρέφει το επώνυμο ενός πελάτη σε μια πληρωμή
	
	· Υλοποίηση μεθόδου που επιστρέφει το όνομα ενός πελάτη σε μια πληρωμή

int PaymentCustomerData::get_codecustomer()
{
return codecustomer;
}
 	
char* PaymentCustomerData::get_lastname()
{
return lastname;
}
 	
char* PaymentCustomerData::get_firstname()
{
return firstname;
}
 	
double PaymentCustomerData::get_amount()
{
return amount;
}
 	
int PaymentCustomerData::get_day()
{
return date.day;
}
 	
int PaymentCustomerData::get_month()
{
return date.month;
}
	· Υλοποίηση μεθόδου που επιστρέφει την αξία πληρωμής ενός πελάτη
	
	· Υλοποίηση μεθόδου που επιστρέφει την ημέρα που έγινε η πληρωμή από ένα πελάτη
	
	· Υλοποίηση μεθόδου που επιστρέφει το μήνα που έγινε η πληρωμή από ένα πελάτη
	
	· Υλοποίηση μεθόδου που επιστρέφει το έτος που έγινε η πληρωμή από ένα πελάτη

int PaymentCustomerData::get_year()
{
return date.year;
}	

2.3.7 Αρχείο paymentsupplier.cpp με υλοποίηση των μεθόδων κλάσης PaymentsupplierData

PaymentSupplierData::PaymentSupplierData(int cn, char l[],char f[],double am, int day, int month, int year)
{
codesupplier=cn;
strcpy(lastname,l);
strcpy(firstname,f);
amount=am;
date.day=day;
date.month=month;
date.year=year;
}
	· Δημιουργός της κλάσης PaymenteSupplierData που θέτει τα ορίσματα (τυπικές παραμέτρους) που λαμβάνει στα μέλη της κλάσης

void PaymentSupplierData::set_codesupplier (int cn)
{
codesupplier=cn;
}
	· Υλοποίηση μεθόδου που τροποποιεί τον κωδικό ενός προμηθευτή σε μια πληρωμή

void PaymentSupplierData::set_lastname (char l[])	· Υλοποίηση μεθόδου που τροποποιεί το επώνυμο ενός προμηθευτή σε μια πληρωμή · Υλοποίηση μεθόδου που τροποποιεί το όνομα ενός προμηθευτή σε μια

{
strcpy(lastname,l);
}
 	
void PaymentSupplierData::set_firstname (char f[])
{
strcpy(firstname,f);
}
 	
void PaymentSupplierData::set_amount (double am)
{
amount=am;
}
 	
void PaymentSupplierData::set_date (int day, int month, int year)
{
date.day=day;
 	
date.month=month;
 	
date.year=year;
}
 	
int PaymentSupplierData::get_codesupplier()
{
return codesupplier;
}
 	
char* PaymentSupplierData::get_lastname()
	πληρωμή
	
	· Υλοποίηση μεθόδου που τροποποιεί την αξία πληρωμής σε ένα προμηθευτή
	
	· Υλοποίηση μεθόδου που τροποποιεί την ημερομηνία πληρωμής σε ένα προμηθευτή
	
	· Υλοποίηση μεθόδου που επιστρέφει τον κωδικό ενός προμηθευτή σε μια πληρωμή
	
	· Υλοποίηση μεθόδου που επιστρέφει το επώνυμο ενός προμηθευτή σε μια πληρωμή
	
	· Υλοποίηση μεθόδου που επιστρέφει το όνομα ενός προμηθευτή σε μια πληρωμή
	
	· Υλοποίηση μεθόδου που επιστρέφει την το ποσό πληρωμής σε ένα προμηθευτή
	
	· Υλοποίηση μεθόδου που επιστρέφει

{
return lastname;
}
 	
char* PaymentSupplierData::get_firstname()
{
return firstname;
}
 	
double PaymentSupplierData::get_amount()
{
return amount;
}
 	
int PaymentSupplierData::get_day()
{
return date.day;
}
 	
int PaymentSupplierData::get_month()
{
return date.month;
}
 	
int PaymentSupplierData::get_year()
{
return date.year;
}
	την ημέρα πληρωμής σε ένα προμηθευτή
	
	· Υλοποίηση μεθόδου που επιστρέφει το μήνα πληρωμής σε ένα προμηθευτή
	
	· Υλοποίηση μεθόδου που επιστρέφει το έτος πληρωμής σε ένα προμηθευτή

2.3.8 Αρχείο supplier_functions.cpp

Όλες οι συναρτήσεις που καλούνται από το υπομενού των προμηθευτών βρίσκονται μέσα στο αρχείο supplier_functionss.cpp και εξηγούνται στον ακόλουθο πίνακα

void cls()	Καθαρισμός οθόνης
void insert_salesupplier(),	Εισαγωγή – καταχώριση αγοράς από προμηθευτή
void insert_supplier()	Εισαγωγή – καταχώριση νέου προμηθευτή
void delete_suppliers()	Διαγραφή όλων των προμηθευτών
void print_suppliers()	Εκτύπωση όλων των προμηθευτών
void update_supplier(int)	Ενημέρωση τηλεφώνου ή διεύθυνσης προμηθευτή
void delete_supplier(int)	Διαγραφή ενός συγκεκριμένου προμηθευτή
void update_supplier_balance(int, double)	Ενημέρωση υπολοίπου ενός συγκεκριμένου προμηθευτή
void print_supplier_sales(int)	Εμφάνιση όλων των αγορών από συγκεκριμένο προμηθευτή
void print_supplier_payments(int)	Εμφάνιση όλων των πληρωμών σε συγκεκριμένο προμηθευτή
void insert_paymentsupplier()	Εισαγωγή – καταχώριση πληρωμής σε προμηθευτή
void print_suppliers_balance(int)	Εμφάνιση υπολοίπου ενός συγκεκριμένου προμηθευτή

void total_average_sales_per_supplier()	Εμφάνιση συγκεντρωτικών πληροφοριών ανά προμηθευτή (π.χ. πλήθος αγορών, συνολική δαπάνη αγορών κ.λ.π.)
void update_items_rest_buing(int, int)	Η συνάρτηση αυτή αυξάνει το υπόλοιπο ενός είδους κατά την ποσότητα αγοράς. Εκτελείται αυτόματα κάθε φορά που γίνεται αγορά από προμηθευτή ώστε να ενημερώσει το υπόλοιπο του είδους

2.3.9 Αρχείο client_functions.cpp

Όλες οι συναρτήσεις που καλούνται από το υπομενού των πελατών βρίσκονται μέσα στο αρχείο client_functionss.cpp και εξηγούνται στον ακόλουθο πίνακα

void cls()	Καθαρισμός οθόνης
void insert_salecustomer(),	Εισαγωγή –καταχώριση πώλησης σε πελάτη
void insert_client()	Εισαγωγή –καταχώριση νέου πελάτη
void delete_clients()	Διαγραφή όλων των πελατών
void print_clients()	Εκτύπωση των στοιχείων όλων των πελατών
void update_client(int)	Ενημέρωση τηλεφώνου ή διεύθυνσης πελάτη
void delete_client(int)	Διαγραφή ενός συγκεκριμένου πελάτη
void update_clients_balance(int, double)	Ενημέρωση υπολοίπου ενός συγκεκριμένου πελάτη
void print_client_sales(int)	Εμφάνιση όλων των πωλήσεων σε συγκεκριμένο πελάτη
void print_client_payments(int)	Εμφάνιση όλων των πληρωμών συγκεκριμένου πελάτη
void insert_paymentcustomer()	Εισαγωγή – καταχώριση πληρωμής πελάτη

void print_clients_balance(int)	Εμφάνιση υπολοίπου ενός συγκεκριμένου πελάτη
void total_average_sales_per_client()	Εμφάνιση συγκεντρωτικών πληροφοριών ανά πελάτη (π.χ. πλήθος πωλήσεων, συνολικό ποσό πληρωμής κ.λ.π.)
void update_items_rest(int, int)	Η συνάρτηση αυτή μειώνει το υπόλοιπο ενός είδους κατά την ποσότητα πώλησης. Εκτελείται αυτόματα κάθε φορά που γίνεται πώληση σε πελάτη ώστε να ενημερώσει το υπόλοιπο του είδους

3 Συμπεράσματα

Το πληροφοριακό σύστημα που κατασκευάσαμε πιστεύουμε ότι απλοποιεί σε πολύ μεγάλο βαθμό τις λειτουργίες μια εμπορικής επιχείρησης που βασίζεται στα 3 δομικά στοιχεία: πελάτες, προμηθευτές και είδη διότι προσφέρει στο χρήστη τις ακόλουθες δυνατότητες:

- Κεντρικό μενού που του δίνει τη δυνατότητα να επιλέξει αρχικά με ποια βασική οντότητα θέλει να εργαστεί (πελάτη, προμηθευτή ή είδος)
- 3 δευτερεύοντα μενού τα οποία συμπεριλαμβάνουν όλες τις πληροφορίες που αφορούν μια οντότητα. Για παράδειγμα αν ο χρήστης επιλέξει να εργαστεί με κάποιο πελάτη, μέσα από το μενού πελατών (Customer's Menu) μπορεί να κάνει οποιαδήποτε εργασία αφορά πελάτη όπως π.χ. καταχώριση νέου πελάτη, τροποποίηση των στοιχείων του, διαγραφή, εισαγωγή πώλησης στον πελάτη αυτό με ταυτόχρονη (δυναμική) ενημέρωση της αποθήκης για το υπόλοιπο του είδους που απομένει μετά την πώληση, εισαγωγή μιας πληρωμής από τον πελάτη με ταυτόχρονη ενημέρωση το υπολοίπου του, εμφάνιση στατιστικών πληροφοριών για κάποιο συγκεκριμένο πελάτη όπως π.χ. πλήθους πωλήσεων που έχει πραγματοποιήσει, συνολικού ποσού που έχει δαπανήσει για τις αγορές αυτές, εμφάνιση του υπολοίπου του λογαριασμού του, εμφάνιση πληροφοριών για όλους τους πελάτες κ.λ.π. Αυτό που αυξάνει τη λειτουργικότητα του προγράμματος αυτού είναι ότι ο χρήστης εκτελεί όλες αυτές τις λειτουργίες μέσα από ένα κεντρικό μενού στο οποίο επανέρχεται μετά την ολοκλήρωση κάθε λειτουργίας ώστε να επιλέξει μια νέα λειτουργία. Οι ίδιες ακριβώς δυνατότητες προσφέρονται στο χρήστη για προμηθευτές και είδη μέσα από αντίστοιχα υπομενού.
- Επιπλέον το πρόγραμμα προσφέρει και μερικές επιπλέον λειτουργίες όπως π.χ. ο έλεγχος του κωδικού κάθε νέου πελάτη, προμηθευτή και είδους ώστε να αποφευχθεί η καταχώριση στοιχείων με τον ίδιο κωδικό. Μιλώντας με όρους των σχεσιακών βάσεων δεδομένων ο κωδικός κάθε πελάτη, προμηθευτή και είδους αποτελεί το πρωτεύον κλειδί της αντίστοιχης οντότητας οπότε πρέπει να είναι μοναδικό γα να διαχωρίζει τις εγγραφές κάθε οντότητας κατά μοναδικό τρόπο.
- Συνοπτικά όλες οι λειτουργίες που προσφέρει το πρόγραμμα αυτό αποτελούν χαρακτηριστικά λειτουργίας της πλειονότητας των προγραμμάτων εμπορικής διαχείρισης που χρησιμοποιούνται σήμερα από τις ελληνικές επιχειρήσεις. Βέβαια

στο πρόγραμμα μπορούν να προστεθούν ακόμα περισσότερες δυνατότητες και λειτουργίες ώστε να καλύψει μεγαλύτερο αριθμό επιχειρήσεων και μεγαλύτερες απαιτήσεις από αυτές. Αυτό μπορεί να αποτελέσει αντικείμενο μελέτης κάποιας επόμενης εργασίας που μπορεί να βελτιώσει και να επεκτείνει την τρέχουσα

3.1 Date.h

```
#ifndef DATE_H

#define DATE_H

class Date //Κλάση περιγραφής ημερομηνίας.

{

public:

        int day;

        int month;

        int year;

};


#endif
```

3.2 clients.h

```
#ifndef CLIENTS_H//Για να αποφύγουμε τη συμπερίληψη του ίδιου header file πολλές φορές
#define CLIENTS_H

#include <iostream>
using std::string;

class ClientData
{
    //Τα επόμενα ιδιωτικά μελη περιγραφουν τα στοιχεία ενός πελατη της της επιχείρησης
    private:
        int codecustomer;
        char lastname[30];
        char firstname[20];
        char adresscode[30];
        char telephonenumber[10];
        double balance;

    public:
        ClientData(int=0,char[]="",char[]="",char[]="",char[]="",double=0.0);

        void set_codecustomer(int);
        void set_lastname(char[]);
        void set_firstname(char[]);
        void set_adresscode(char[]);
        void set_telephonenumber(char[]);
        void set_balance(double);
```

```
        //Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης
        int get_codecustomer();
        char* get_lastname();
        char* get_firstname();
        char* get_adresscode();
        char* get_telephonenumber();
        double get_balance();
};
#endif
```

3.3 Suppliers.h

```
#ifndef SUPPLIERS_H//Για να αποφύγουμε τη συμπερίληψη του ίδιου header file πολλές φορές
#define SUPPLIERS_H

#include <iostream>
using std::string;

class SupplierData
{
	//Τα επόμενα ιδιωτικά μελη περιγραφουν τα στοιχεία ενός προμηθευτή της επιχείρησης
	private:
		int codesupplier;
		char lastname[30];
		char firstname[20];
		char adresscode[30];
		char telephonenumber[10];
		double balance;

	public:
		SupplierData(int=0,char[]="",char[]="",char[]="",char[]="",double=0.0);
		void set_codesupplier(int);
		void set_lastname(char[]);
		void set_firstname(char[]);
		void set_adresscode(char[]);
		void set_telephonenumber(char[]);
		void set_balance(double);

		//Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης
```

```
        int get_codesupplier();
        char* get_lastname();
        char* get_firstname();
        char* get_adresscode();
        char* get_telephonenumber();
        double get_balance();
};
#endif
```

3.4 Items.h

```
#ifndef ITEMS_H//Για να αποφύγουμε τη συμπερίληψη του ίδιου header file πολλές φορές
#define ITEMS_H

#include <iostream>
using std::string;

class ItemData
{
	//Τα επόμενα ιδιωτικά μελη περιγραφουν τσ στοιχεία ενός είδους της επιχείρησης
	private:
		int itemcode;
		char itemname[30];
		double cost;
		int rest;//υπολοιπο ειδους
	public:
		ItemData(int=0,char[]="",double=0.0,int=0);
		void set_itemcode(int);
		void set_itemname(char[]);
		void set_cost(double);
		void set_rest(int);

		//Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης
		int get_itemcode();
		char* get_itemname();
		double get_cost();
		int get_rest();
};
#endif
```

3.5 Paymentcustomer.h

```
#ifndef PAYMENTCUSTOMER_H
#define PAYMENTCUSTOMER_H

#include <iostream>
using std::string;

#include "date.h"

class PaymentCustomerData
{
	private:
		int codecustomer;
		char lastname[30];
		char firstname[20];
		double amount;//ποσο πληρωμης πελατη
		Date date;//Ημερομηνια πληρωμης πελατη

	public:

	PaymentCustomerData(int=0,char[]="",char[]="",double=0.0,int=0,int=0,int=0);
		void set_codecustomer(int);
		void set_lastname(char[]);
		void set_firstname(char[]);
		void set_amount(double);
		void set_date(int,int,int);

		//Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης
		int get_codecustomer();
		char* get_lastname();
```

```
        char* get_firstname();
        double get_amount();
        int get_day();
        int get_month();
        int get_year();
};
#endif
```

3.6 Paymentsupplier.h

```
#ifndef PAYMENTSUPPLIER_H
#define PAYMENTSUPPLIER_H
#include <iostream>
using std::string;

#include "date.h"

class PaymentSupplierData
{
	private:
		int codesupplier;
		char lastname[30];
		char firstname[20];
		double amount;//ποσο πληρωμης προμηθευτη
		Date date;//Ημερομηνια πληρωμης προμηθευτη

	public:

	PaymentSupplierData(int=0,char[]="",char[]="",double=0.0,int=0,int=0,int=0);
		void set_codesupplier(int);
		void set_lastname(char[]);
		void set_firstname(char[]);
		void set_amount(double);
		void set_date(int,int,int);

		//Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης
		int get_codesupplier();
		char* get_lastname();
		char* get_firstname();
```

```
        double get_amount();
        int get_day();
        int get_month();
        int get_year();
};
#endif
```

3.7 Salecustomer.h

```
#ifndef SALECUSTOMER_H//Για να αποφύγουμε τη συμπερίληψη του ίδιου header file πολλές φορές
#define SALECUSTOMER_H

#include <iostream>
using std::string;

#include "date.h"

class SaleCustomerData
{
	//Τα 4 επόμενα ιδιωτικά μελη περιγραφουν τσ στοιχεία ενός πελατητης της επιχείρησης
	private:
		int codecustomer;
		char lastname[30];
		char firstname[20];
		double amount;//αξια πωλησης πελατη
		Date date;//Ημερομηνια κινησης πελατη
		int itemcode;//Κωδικός είδους που αγοράζει ο πελάτης
		int quantity;//Ποσότητα αγοράς

	public:

	SaleCustomerData(int=0,char[]="",char[]="",double=0.0,int=0,int=0,int=0,int=0,int=0);
		void set_codecustomer(int);
		void set_lastname(char[]);
		void set_firstname(char[]);
		void set_amount(double);
```

```
        void set_date(int,int,int);
        void set_itemcode(int);
        void set_quantity(int);

        //Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης
        int get_codecustomer();
        char* get_lastname();
        char* get_firstname();
        double get_amount();
        int get_itemcode();
        int get_quantity();

        int get_day();
        int get_month();
        int get_year();
};
#endif
```

3.8 Salesupplier.h

```
#ifndef SALESUPPLIER_H//Για να αποφύγουμε τη συμπερίληψη του ίδιου header file πολλές φορές
#define SALESUPPLIER_H

#include <iostream>
using std::string;

#include "date.h"

class SaleSupplierData
{
	//Τα 4 επόμενα ιδιωτικά μελη περιγραφουν τσ στοιχεία ενός πελατητης της επιχείρησης
	private:
		int codesupplier;
		char lastname[30];
		char firstname[20];
		double amount;//αξια πωλησης προμηθευτη
		Date date;//Ημερομηνια κινησης προμηθευτη
		int itemcode;//Κωδικός είδους που αγοράζει η επειχειπηση απο τον προμηθευτη
		int quantity;//Ποσότητα αγοράς

	public:

	SaleSupplierData(int=0,char[]="",char[]="",double=0.0,int=0,int=0,int=0,int=0);
		void set_codesupplier(int);
		void set_lastname(char[]);
		void set_firstname(char[]);
		void set_amount(double);
```

```
        void set_date(int,int,int);
        void set_itemcode(int);
        void set_quantity(int);

        //Μέθοδοι get που επιστρέφουν τις τιμές των ιδιωτικών μελών της κλάσης
        int get_codesupplier();
        char* get_lastname();
        char* get_firstname();
        double get_amount();
        int get_itemcode();
        int get_quantity();


        int get_day();
        int get_month();
        int get_year();
};
#endif
```

3.9 clients.cpp

```
#include <iostream>

using std::string;


#include <cstring>


#include "epixeir.h"//αυτό το hader file δεν βρίσκεται στο φάκελο που είναι όλα τα header files αλλά στον τρέχοντα φάκελο. Γιαυτό το όνομα του μπαίνει σε ""


ClientData::ClientData(int c,char l[],char f[],char a[],char t[],double b)

{

	codecustomer=c;


	strcpy(lastname,l);


	strcpy(firstname,f);


	strcpy(adresscode,a);


	strcpy(telephonenumber,t);


	balance=b;

}



void ClientData::set_codecustomer(int c)

{

	codecustomer=c;

}
```

```
void ClientData::set_lastname(char l[])
{
	strcpy(lastname,l);
}

void ClientData::set_firstname(char f[])
{
	strcpy(firstname,f);
}

void ClientData::set_adresscode(char a[])
{
	strcpy(adresscode,a);
}

void ClientData::set_telephonenumber(char t[])
{
	strcpy(telephonenumber,t);
}

void ClientData::set_balance(double b)
{
	balance=b;
}


int ClientData::get_codecustomer()
{
	return codecustomer;
}
```

```
char* ClientData::get_lastname()
{
	return lastname;
}

char* ClientData::get_firstname()
{
	return firstname;
}

char* ClientData::get_adresscode()
{
	return adresscode;
}
char* ClientData::get_telephonenumber()
{
	return telephonenumber;
}

double ClientData::get_balance()
{
	return balance;
}
```

3.10 Client_functions.cpp

```
#include <iostream>//βιβλιοθηκη με συναρτησεις I/O

using std::cin;

using std::cout;

using std::endl;

using std::ios;

using std::cerr;

using std::left;

using std::right;

using std::fixed;


#include <fstream>//βιβλιοθηκη με συναρτησεις διαχειρισεις αρχειων

using std::ofstream;//τυπος για καταχωριση σε αρχειο (αρχεία εξόδου)

using std::ifstream;//τυπος για διάβασμα απο αρχειο (αρχεία εισόδου)

using std::fstream;//τυπος για διάβασμα απο αρχειο και εκτυπωση σε αρχείο (αρχεία εισόδου-εξόδου)


#include <cstdlib>


#include <iomanip>//περιεχει τις συναρτησεις setw ,setprecision.

using std::setw;

using std::setprecision;


void
cls(),delete_clients(),insert_client(),insert_salecustomer(),print_clients(),update_client(int),delete_client(int),update_clients_balance(int,double),update_supplier_balance(int,double),update_items_rest(int,int),print_client_sales(int),print_clients_balance(int),total_average_sales_per_customer(),insert_paymentcustomer(),print_client_payments(int);



#include "epixeir.h"
```

```
void delete_clients()
{
        char ans[10];

        cout<<"Attention!!"<<endl;
        cout<<"Really delete file of customers y/n"<<endl;
        cin>>ans;

        if (strcmp(ans,"y")==0 || strcmp(ans,"Y")==0)
        {
                ofstream acf("customer.txt");
                ofstream acf1("paymentcustomer.txt");
                ofstream acf2("salecustomer.txt");
   }
}

void insert_client()
{
        int r,cust;
        char fn[20],ln[30],adrc[30],teln[10];
        bool found;

        ClientData client;

        cout<<setw(20)<<"INSERT NEW CUSTOMER"<<endl;
        for (r=0;r<20;r++)
                cout<<"-";

        do
        {
```

```
        ifstream acfr("customer.txt");

        found=false;

        cout<<endl<<"Enter Customer's Code: "<<endl;
        cin>>cust;

        acfr.read(reinterpret_cast<char*>(&client),sizeof(ClientData));

        while(!acfr.eof())
        {
                if (client.get_codecustomer()==cust)
                {
                        found=true;
                        break;
                }

                acfr.read(reinterpret_cast<char*>(&client),sizeof(ClientData));
        }

        acfr.close();//κλεινουμε το αρχειο πελατων

        if (found==true)
                cout<<"Customer already exists"<<endl;
}
while (found==true);

client.set_codecustomer(cust);

ofstream acf("customer.txt",ios::app);
```

```
	cout<<"Enter Customer's First Name: "<<endl;
	cin>>fn;
	client.set_firstname(fn);

	cout<<"Enter Customer's Last Name: "<<endl;
	cin>>ln;
	client.set_lastname(ln);

	cout<<"Enter Customer's Address Code: "<<endl;
	cin>>adrc;

	client.set_adresscode(adrc);

	cout<<"Enter Customer's Telephone Number: "<<endl;
	cin>>teln;
	client.set_telephonenumber(teln);

	client.set_balance(0);

	acf.write(reinterpret_cast<const char*>(&client),sizeof(ClientData));

	acf.close();//κλεινουμε το αρχειο πελατων
}


void insert_salecustomer()
{
	int r,cn,day,month,year,pr,q,ypoloipo;
	double tm;
```

```
bool found=false;

ofstream acf("salecustomer.txt",ios::app);
SaleCustomerData salecust;

if (acf.fail()==true)
{
        cout<<"Sales Customer File does not exists!"<<endl;
        ofstream acf("salecustomer.txt");
        return;
}

cout<<setw(20)<<"INSERT NEW CUSTOMER SALE "<<endl;
for (r=0;r<20;r++)
        cout<<"-";

cout<<endl<<"Enter The Customer's Code: "<<endl;
cin>>cn;

ifstream acfc("customer.txt",ios::in);
ClientData client;

acfc.read(reinterpret_cast<char*>(&client),sizeof(ClientData));

while(!acfc.eof())
{
        if (client.get_codecustomer()!=0)
                if (cn==client.get_codecustomer())
                {
                        salecust.set_codecustomer(cn);
```

```
                    salecust.set_firstname(client.get_firstname());
                    salecust.set_lastname(client.get_lastname());
                    found=true;
                }
        acfc.read(reinterpret_cast<char*>(&client),sizeof(ClientData));
}

acfc.close();//κλεινουμε το αρχειο πελατων

if (found==false)
{
        cout<<"Customer doesn't exists!"<<endl;
        return;
}

cout<<"Enter Product Code "<<endl;
cin>>pr;

ifstream acfp("items.txt",ios::in);

if (acfp.fail()==true)
{
        cout<<"Items File not exists!"<<endl;
        ofstream acf("items.txt");
        return;
}

ItemData itt;

acfp.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));
```

```
found=false;

while(!acfp.eof())
{
	if (itt.get_itemcode()!=0)
		if (pr==itt.get_itemcode())
		{
			salecust.set_itemcode(pr);
			ypoloipo=itt.get_rest();
			tm=itt.get_cost();
			found=true;
		}

	acfp.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));
}

acfp.close();//κλεινουμε το αρχειο ειδων

if (found==false)
{
	cout<<"Item doesn't exists!"<<endl;
	return;
}

if (ypoloipo==0)
{
	cout<<"There is not any Quantity of this product!!!"<<endl;
	cout<<"Buying of this product is impossible!!!"<<endl;
	return;
```

```
}

do
{
        cout<<"Enter Product Quantity "<<endl;
        cin>>q;

        if (q>ypoloipo)
                cout<<"There is not enough Quantity of this product!!!"<<endl;
}
while (q>ypoloipo);

salecust.set_amount(q*tm);

cout<<"Enter The Day/Month/Year Of Sale"<<endl;
do
{
        cin>>day;
}
while(day<1 || day>31);

do
{
        cin>>month;
}
while(month<1 ||month>12);

do
{
        cin>>year;
```

```
	}
	while(year<1998 || year>3000);

	salecust.set_quantity(q);
	salecust.set_amount(q*tm);
	salecust.set_date(day,month,year);

	acf.write(reinterpret_cast<const char*>(&salecust),sizeof(SaleCustomerData));

	acf.close();//κλεινουμε το αρχειο πωλήσεων

	update_clients_balance(cn,q*tm);

	update_items_rest(pr,q);
}


void insert_paymentcustomer()
{
	int r,cn,day,month,year;
	double amount;
	bool found=false;

	ofstream acf("paymentcustomer.txt",ios::app);
	PaymentCustomerData paycust;

	cout<<setw(20)<<"INSERT NEW CUSTOMER PAYMENT "<<endl;
	for (r=0;r<20;r++)
		cout<<"-";
```

```
cout<<endl<<"Enter The Customer's Code: "<<endl;
cin>>cn;

ifstream acfc("customer.txt",ios::in);
ClientData client;

acfc.read(reinterpret_cast<char*>(&client),sizeof(ClientData));

while(!acfc.eof())
{
        if (client.get_codecustomer()!=0)
                if (cn==client.get_codecustomer())
                {
                        paycust.set_codecustomer(cn);
                        paycust.set_firstname(client.get_firstname());
                        paycust.set_lastname(client.get_lastname());
                        found=true;
                }
        acfc.read(reinterpret_cast<char*>(&client),sizeof(ClientData));
}

acfc.close();//κλεινουμε το αρχειο πελατων

if (found==false)
{
        cout<<"Customer doesn't exists!"<<endl;
        return;
}

cout<<"Enter The Day/Month/Year of Payment"<<endl;
```

```
do
{
	cin>>day;

	if (day<1 || day>31)
		cout<<"Wrong Day. Enter another day from 1-31"<<endl;
}
while (day<1 || day>31);

do
{
	cin>>month;

	if (month<1 || month>12)
		cout<<"Wrong Month. Enter another month from 1-12"<<endl;

}
while(month<1 ||month>12);

do
{
	cin>>year;

	if (year<1998 || year>3000)
		cout<<"Wrong Year. Enter another year from 1998-2100"<<endl;

}
while(year<1998 || year>2100);

cout<<"Enter payments amount"<<endl;
```

```
	cin>>amount;

	paycust.set_amount(amount);
	paycust.set_date(day,month,year);

	acf.write(reinterpret_cast<const char*>(&paycust),sizeof(PaymentCustomerData));

	acf.close();//κλεινουμε το αρχειο πωλήσεων

	update_clients_balance(cn,-amount);
}


void print_client_payments(int cust)
{
	ifstream acf("paymentcustomer.txt");
	PaymentCustomerData paycust;
	int r;

	if (acf.fail()==true)
	{
		cout<<"Customer's Payments File does not exists!"<<endl;
		ofstream acf("paymentcustomer.txt");
		return;
	}

	cout<<setw(39)<<"CUSTOMER PAYMENTS"<<endl;
	for (r=0;r<77;r++)
		cout<<"-";
```

```
	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(9)<<"AMOUNT"<<right<<setw(12)<<"DATE"<<endl;


	acf.read(reinterpret_cast<char*>(&paycust),sizeof(PaymentCustomerData));


	while(!acf.eof())
	{
		if (paycust.get_codecustomer()==cust)

	cout<<left<<setw(7)<<paycust.get_codecustomer()<<setw(13)<<paycust.get_lastnam
e()<<setw(15)<<paycust.get_firstname()<<setw(9)<<paycust.get_amount()<<setw(7)<<right
<<paycust.get_day()<<"/"<<paycust.get_month()<<"/"<<paycust.get_year()<<endl;


		acf.read(reinterpret_cast<char*>(&paycust),sizeof(PaymentCustomerData));
	}


	acf.close();//κλεινουμε το αρχειο πληρωμων


	for (r=0;r<77;r++)
		cout<<"-";
	cout<<endl<<endl;
}



void update_client(int cust)
{
	bool found=false;
	char type;
	char newaddr[30],newtel[10];
	int r,i=0;


	fstream acf("customer.txt",ios::in|ios::out);
```

```
ClientData client;

cout<<setw(20)<<"UPDATE CUSTOMER"<<endl;
for (r=0;r<20;r++)
        cout<<"-";

while(!acf.eof())
{
        acf.read(reinterpret_cast<char*>(&client),sizeof(ClientData));

        i++;

        if (client.get_codecustomer()==cust)
        {
                found=true;
                break;
        }
}

if(found==false)
{
        cout<<"Customer Code "<<cust<<" does not exist"<<endl;
        return;
}

cout<<endl<<"Client exists and is the following one:"<<endl;
cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(18)<<"ADDRESS"<<setw(17)<<"TELEPHONE"<<setw(15)<<"BAL
ANCE"<<endl;
cout<<left<<setw(7)<<client.get_codecustomer()<<setw(13)<<client.get_lastname()<
<setw(15)<<client.get_firstname()<<setw(18)<<client.get_adresscode()<<setw(17)<<client.g
et_telephonenumber()<<right<<setw(7)<<client.get_balance()<<endl;
```

```
cout<<"Update Customer's (A)ddress or (T)elephone?"<<endl;
cin>>type;

if (type=='a' || type=='A')
{
        cout<<"Enter new Customer's Address"<<endl;
        cin>>newaddr;

        client.set_adresscode(newaddr);

        cout<<"Client's address succesfully changed!"<<endl;

}

if (type=='t' || type=='T')
{
        cout<<"Enter new Customer's Telephone"<<endl;
        cin>>newtel;

        client.set_telephonenumber(newtel);

        cout<<"Client's telephone succesfully changed!"<<endl;

}
```

acf.seekg((i-1)*sizeof(ClientData));//Η συναρτηση seekg Μεταφερει το δεικτη του αρχειου σε μια συγκεκριμενη θεση.Εδω μεταφερουμε

//το δεικτη του αρχειου στη προηγουμενη εγγραφη αυτης που θελουμε να ενημεροσουμε το υπολοιπο

```
	acf.write(reinterpret_cast<char*>(&client),sizeof(ClientData));

	acf.close();//κλεινουμε το αρχειο πελατών
}

void total_average_sales_per_customer()
{
	bool found=false;
	int i=0,j=0,n,pol=0;
	double sum=0;

	fstream acf1("customer.txt",ios::in),acf2("customer.txt",ios::in);
	ClientData client;

	while(!acf1.eof())
	{
		acf1.read(reinterpret_cast<char*>(&client),sizeof(ClientData));

		i++;
	}
	acf1.close();

	n=i-1;//Υπολογίζουμε το πλήθος των εγγραφών του αρχείου
	ClientData *cd=new ClientData[n];//Δεσμευουμε δυναμικα μνημη για να τις τοποθετησουμε σε πίνακα

	while(!acf2.eof())
		acf2.read(reinterpret_cast<char*>(&cd[j++]),sizeof(ClientData));//διαβαζουμε μια εγγραφή πελάτη και τη βάζοθμε στο πίνακα πελατών
```

```
SaleCustomerData salecust;

cout<<setw(39)<<"TOTAL & AVERAGE  SALES PER CUSTOMER"<<endl;

for (i=0;i<n;i++)
{
        ifstream acf3("salecustomer.txt");

        sum=0;
        pol=0;

        acf3.read(reinterpret_cast<char*>(&salecust),sizeof(SaleCustomerData));

        while(!acf3.eof())
        {
                if (salecust.get_codecustomer()==cd[i].get_codecustomer())
                {
                        sum+=salecust.get_amount();
                        pol++;
                }

acf3.read(reinterpret_cast<char*>(&salecust),sizeof(SaleCustomerData));
        }

cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(9)<<"SALES#"<<setw(11)<<"TOTAL AMOUNT"<<setw(20)<<"
AVG AMOUNT/SALE"<<endl;

cout<<setw(7)<<cd[i].get_codecustomer()<<setw(13)<<cd[i].get_lastname()<<setw(
15)<<cd[i].get_firstname()<<setw(11)<<pol<<setw(11)<<sum<<setw(20)<<sum/pol<<endl;
```

```
		cout<<endl;
		acf3.close();//κλεινουμε το αρχειο πωλήσεων
	}
}

void print_clients_balance(int custbal)
{
	ifstream acf("customer.txt");
	ClientData client;
	int r;
	bool found=false;

	if (acf.fail()==true)
	{
		cout<<"Customer's File does not exists!"<<endl;
		ofstream acf("customer.txt");
		return;
	}

	cout<<setw(39)<<"CUSTOMER BALANCE"<<endl;
	for (r=0;r<43;r++)
		cout<<"-";

	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(18)<<setw(15)<<"BALANCE"<<endl;

	acf.read(reinterpret_cast<char*>(&client),sizeof(ClientData));

	while(!acf.eof())
	{
		if (client.get_codecustomer()==custbal)
```

```
                {

        cout<<left<<setw(7)<<client.get_codecustomer()<<setw(13)<<client.get_lastname()<
<setw(15)<<client.get_firstname()<<setw(18)<<right<<setw(7)<<client.get_balance()<<endl
;
                        found=true;
                        break;
                }

                acf.read(reinterpret_cast<char*>(&client),sizeof(ClientData));
        }

        acf.close();//κλεινουμε το αρχειο πωλήσεων

        for (r=0;r<43;r++)
                cout<<"-";
        cout<<endl<<endl;

        if (found==false)
                cout<<"Client does not exists"<<endl;
}



void print_clients()
{
        ifstream acf("customer.txt",ios::in);

        if (acf.fail()==true)
        {
                cout<<"Clients File not exists!"<<endl;
                ofstream acf("customer.txt");
```

```
		return;
	}

	ClientData client;
	int r;

	cout<<setw(39)<<"CUSTOMERS"<<endl;
	for (r=0;r<77;r++)
		cout<<"-";
	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(18)<<"ADDRESS"<<setw(17)<<"TELEPHONE"<<setw(15)<<"BAL
ANCE"<<endl;

	acf.read(reinterpret_cast<char*>(&client),sizeof(ClientData));

	while(!acf.eof())
	{
		if (client.get_codecustomer()!=0)

	cout<<left<<setw(7)<<client.get_codecustomer()<<setw(13)<<client.get_lastname()<
<setw(15)<<client.get_firstname()<<setw(18)<<client.get_adresscode()<<setw(17)<<client.g
et_telephonenumber()<<right<<setw(7)<<client.get_balance()<<endl;

		acf.read(reinterpret_cast<char*>(&client),sizeof(ClientData));
	}

	acf.close();//κλεινουμε το αρχειο πελατων

	for (r=0;r<77;r++)
		cout<<"-";
	cout<<endl<<endl;
}
```

```
void print_client_sales(int cust)
{
	ifstream acf("salecustomer.txt");
	SaleCustomerData salecust;
	int r;

	if (acf.fail()==true)
	{
		cout<<"Customer's Sales File does not exists!"<<endl;
		ofstream acf("salecustomer.txt");
		return;
	}

	cout<<setw(39)<<"CUSTOMER SALES"<<endl;
	for (r=0;r<77;r++)
		cout<<"-";

	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(9)<<"ITEM"<<setw(11)<<"QUANTITY"<<setw(18)<<"AMOUNT"<
<setw(8)<<"DATE"<<endl;

	acf.read(reinterpret_cast<char*>(&salecust),sizeof(SaleCustomerData));

	while(!acf.eof())
	{
		if (salecust.get_codecustomer()==cust)

	cout<<left<<setw(7)<<salecust.get_codecustomer()<<setw(13)<<salecust.get_lastna
me()<<setw(15)<<salecust.get_firstname()<<setw(9)<<salecust.get_itemcode()<<setw(11)<
<salecust.get_quantity()<<right<<setw(4)<<salecust.get_amount()<<setw(7)<<salecust.get_
day()<<"/"<<salecust.get_month()<<"/"<<salecust.get_year()<<endl;
```

```
            acf.read(reinterpret_cast<char*>(&salecust),sizeof(SaleCustomerData));
        }

        acf.close();//κλεινουμε το αρχειο πωλήσεων

        for (r=0;r<77;r++)
            cout<<"-";
        cout<<endl<<endl;
}

void update_clients_balance(int cust,double am)
{
        bool found=false;
        int i=0;

        fstream acf("customer.txt",ios::in|ios::out);
        ClientData client;

        while(!acf.eof())
        {
            acf.read(reinterpret_cast<char*>(&client),sizeof(ClientData));

            i++;

            if (client.get_codecustomer()==cust)
            {
                found=true;
                client.set_balance(client.get_balance()+am);
                break;
```

```
                }
        }

        cout<<endl<<"Client with new balance after new buying of  amount "<<am<<"
is:"<<endl;
        cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(18)<<"ADDRESS"<<setw(17)<<"TELEPHONE"<<setw(15)<<"BAL
ANCE"<<endl;
        cout<<left<<setw(7)<<client.get_codecustomer()<<setw(13)<<client.get_lastname()<
<setw(15)<<client.get_firstname()<<setw(18)<<client.get_adresscode()<<setw(17)<<client.g
et_telephonenumber()<<right<<setw(7)<<client.get_balance()<<endl;

        acf.seekg((i-1)*sizeof(ClientData));//Η συναρτηση seekg Μεταφερει το δεικτη του
αρχειου σε μια συγκεκριμενη θεση.Εδω μεταφερουμε
                                                                //το δεικτη του
αρχειου στη προηγουμενη εγγραφη αυτης που θελουμε να ενημεροσουμε το υπολοιπο

        acf.write(reinterpret_cast<char*>(&client),sizeof(ClientData));

        acf.close();//κλεινουμε το αρχειο πωλήσεων
}

void delete_client(int cust)
{
        bool found=false;
        char ans;
        int r,i=0,j=0,n,pos,k;

        fstream acf("customer.txt",ios::in|ios::out),acf3("customer.txt",ios::in|ios::out);
        ClientData client;

        cout<<setw(20)<<"DELETE CUSTOMER"<<endl;
        for (r=0;r<20;r++)
```

```
            cout<<"-";


        cout<<endl;


        while(!acf.eof())
        {
            acf.read(reinterpret_cast<char*>(&client),sizeof(ClientData));


            i++;
        }


        n=i-1;//Υπολογίζουμε το πλήθος των εγγραφών του αρχείου
        ClientData *cd=new ClientData[n];//Δεσμευμε δυναμηκα μνημη για να τις τοποθετησουμε σε πίνακα


        while(!acf3.eof())
            acf3.read(reinterpret_cast<char*>(&cd[j++]),sizeof(ClientData));//διαβαζουμε μια εγγραφή προμηθευτη και τη βάζοθμε στο πίνακα προμηθευτων


        for (i=0;i<n;i++)
        {
            if (cd[i].get_codecustomer()==cust)
            {
                pos=i;//Αν ο πελατης υπάρχει βάζουμε  τη θέση του στη μεταβλητή i
                found=true;
                break;
            }
        }


        if(found==false)
        {
```

```
            cout<<"Client Code "<<cust<<" does not exist"<<endl;
            return;
        }

        cout<<endl<<"Client exists and is the following one:"<<endl;
        cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(18)<<"ADDRESS"<<setw(17)<<"TELEPHONE"<<setw(15)<<"BAL
ANCE"<<endl;
        cout<<left<<setw(7)<<cd[pos].get_codecustomer()<<setw(13)<<cd[pos].get_lastnam
e()<<setw(15)<<cd[pos].get_firstname()<<setw(18)<<cd[pos].get_adresscode()<<setw(17)<
<cd[pos].get_telephonenumber()<<right<<setw(7)<<cd[pos].get_balance()<<endl;

        cout<<"DO YOU WANT TO DELETE HIM/HER? (Y/N)"<<endl;
        cin>>ans;

        if (ans=='y' || ans=='Y')
        {
            for (i=pos;i<n-1;i++)//Κάνουμε αριστερή ολισθιση κατα μια θεση στους
πελατες
                cd[i]=cd[i+1];

            n--;//Μειωνουμε το μεγεθος των προμηθευτων κατα 1

            fstream acf2("customer.txt",ios::out);

            for(i=0;i<n;i++)
                acf2.write(reinterpret_cast<char*>(&cd[i]),sizeof(ClientData));

            acf2.close();//κλεινουμε το αρχειο πελατων
            acf.close();//κλεινουμε το αρχειο πελατων

            cout<<"Client with code "<<cust<<" has been succesfully deleted!"<<endl;
```

```
        ifstream acfs("paymentcustomer.txt",ios::in),acf3s("client.txt",ios::in);
    PaymentCustomerData paycust;

        if (acfs.fail()==true)
        {
            cout<<"Payment Client File does not exists!"<<endl;
            ofstream acf("paymentcustomer.txt");
        }

        i=0;

        while(!acfs.eof())
        {

    acfs.read(reinterpret_cast<char*>(&paycust),sizeof(PaymentCustomerData));

            i++;
        }

        n=i-1;//Υπολογίζουμε το πλήθος των εγγραφών του αρχείου

        if (n==0)
            cout<<"There aren't any payments from this Customer"<<endl;
        else
        {
            PaymentCustomerData *cd1=new
PaymentCustomerData[n];//Δεσμεουμε δυναμηκα μνημη για να τις τοποθετησουμε σε
πίνακα
            PaymentCustomerData *newcd1=new
PaymentCustomerData[n];//Δεσμεουμε δυναμηκα μνημη για να τις τοποθετησουμε σε
πίνακα
```

```
                    j=0;

                    while(!acf3s.eof())

acf3s.read(reinterpret_cast<char*>(&cd1[j++]),sizeof(PaymentCustomerData));

                    k=0;

                    for (i=0;i<n;i++)
                            if (cd1[i].get_codecustomer()!=cust)
                                    newcd1[k++]=cd1[i];


                    fstream acf2s("paymentcustomer.txt",ios::out);

                    for(i=0;i<k;i++)

        acf2s.write(reinterpret_cast<char*>(&newcd1[i]),sizeof(PaymentCustomerData));

                    acf2s.close();
                    acfs.close();
                    acf3s.close();


                    cout<<"All Client Payments have been succesfully deleted!"<<endl;
                }

                ifstream acfsc("salecustomer.txt",ios::in),acf3sc("salecustomer.txt",ios::in);
                SaleCustomerData salecust;
```

```
i=0;

while(!acfsc.eof())
{
acfsc.read(reinterpret_cast<char*>(&salecust),sizeof(SaleCustomerData));

i++;
}

n=i-1;//Υπολογίζουμε το πλήθος των εγγραφών του αρχείου

if (n==0)
cout<<"There aren't any sales to this customer"<<endl;
else
{
SaleCustomerData *cd2=new SaleCustomerData[n];//Δεσμεουμε δυναμηκα μνημη για να τις τοποθετησουμε σε πίνακα
SaleCustomerData *newcd2=new SaleCustomerData[n];//Δεσμεουμε δυναμηκα μνημη για να τις τοποθετησουμε σε πίνακα

j=0;

while(!acf3sc.eof())
acf3sc.read(reinterpret_cast<char*>(&cd2[j++]),sizeof(SaleCustomerData));

k=0;

for (i=0;i<n;i++)
if (cd2[i].get_codecustomer()!=cust)
newcd2[k++]=cd2[i];
```

```
                fstream acf2sc("salecustomer.txt",ios::out);

                for(i=0;i<k;i++)

        acf2sc.write(reinterpret_cast<char*>(&newcd2[i]),sizeof(SaleCustomerData));

                acf2sc.close();
                acf3sc.close();
                acfsc.close();

                cout<<"All Customer Sales have been succesfully deleted!"<<endl;
            }
            cout<<"\n 1 \n";
        }

        cout<<"\n 2 \n";

}
```

3.11 Items.cpp

```
#include <iostream>
using std::string;

#include <cstring>

#include "epixeir.h"

ItemData::ItemData(int c,char n[],double k,int r)
{
        itemcode=c;
        strcpy(itemname,n);
        cost=k;
        rest=r;
}



void ItemData::set_itemcode(int c)
{
        itemcode=c;
}

void ItemData::set_itemname(char n[])
{
        strcpy(itemname,n);
}

void ItemData::set_cost(double k)
{
        cost=k;
```

```
}

void ItemData::set_rest(int r)
{
        rest=r;
}

int ItemData::get_itemcode()
{
        return itemcode;
}

char* ItemData::get_itemname()
{
        return itemname;
}

double ItemData::get_cost()
{
        return cost;
}

int ItemData::get_rest()
{
        return rest;
}
```

3.12 Item_function.cpp

```
#include <iostream>//βιβλιοθηκη με συναρτησεις I/O

using std::cin;

using std::cout;

using std::endl;

using std::ios;

using std::cerr;

using std::left;

using std::right;

using std::fixed;


#include <fstream>//βιβλιοθηκη με συναρτησεις διαχειρισεις αρχειων

using std::ofstream;//τυπος για καταχωριση σε αρχειο (αρχεία εξόδου)

using std::ifstream;//τυπος για διάβασμα απο αρχειο (αρχεία εισόδου)

using std::fstream;//τυπος για διάβασμα απο αρχειο και εκτυπωση σε αρχείο (αρχεία εισόδου-εξόδου)


#include <cstdlib>


#include <iomanip>//περιεχει τις συναρτησεις setw ,setprecision.

using std::setw;

using std::setprecision;


void
delete_clients(),insert_salecustomer(),insert_salesupplier(),insert_supplier(),print_clients(),update_client(int),delete_suppliers(),insert_supplier(),print_suppliers(),update_supplier(int),delete_client(int),delete_supplier(int),print_items(),insert_item(),delete_items(),update_item(int),
delete_item(int),update_clients_balance(int,double),update_supplier_balance(int,double),update_items_rest(int,int),update_items_rest_buing(int,int),
print_client_sales(int),print_supplier_sales(int),print_clients_balance(int),total_average_sales_per_customer(),insert_paymentcustomer(),print_client_payments(int),print_supplier_payments(int),insert_paymentsupplier(),print_suppliers_balance(int),total_average_sales_per_supplier(),print_items_rest(int),print_items_in_lack(int),print_item_sales(int),print_item_purchases(int);
```

```
#include "epixeir.h"

void delete_items()
{
	char ans[10];

	cout<<"Attention!!"<<endl;
	cout<<"Really delete file of items y/n"<<endl;
	cin>>ans;

	if (strcmp(ans,"y")==0 || strcmp(ans,"Y")==0)
		ofstream acf("items.txt");
}


void insert_item()
{
	int res,cod,r;
	char name[30];
	double cost;

	ofstream acf("items.txt",ios::app);
	ItemData itt;

	cout<<setw(20)<<"INSERT NEW ITEM"<<endl;
	for (r=0;r<20;r++)
		cout<<"-";

	cout<<endl<<"Enter ITEM's Code: "<<endl;
```

```
	cin>>cod;
	itt.set_itemcode(cod);

	cout<<"Enter ITEM's  Name: "<<endl;
	cin>>name;
	itt.set_itemname(name);

	cout<<"Enter ITEM's cost: "<<endl;
	cin>>cost;
	itt.set_cost(cost);

	cout<<endl<<"Enter ITEM's REst: "<<endl;
	cin>>res;
	itt.set_rest(res);

	acf.write(reinterpret_cast<const char*>(&itt),sizeof(ItemData));

	acf.close();//κλεινουμε το αρχειο πελατων
}

void print_items()
{
	ifstream acf("items.txt",ios::in);
	ItemData itt;
	int r;

	cout<<setw(39)<<"ITEMS"<<endl;
	for (r=0;r<77;r++)
		cout<<"-";
```

```
	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"ITEMNAME"<<setw(15)<<"
COST/ITEM"<<setw(15)<<"REST"<<endl;


	acf.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));


	while(!acf.eof())
	{
		if (itt.get_itemcode()!=0)

	cout<<left<<setw(7)<<itt.get_itemcode()<<setw(13)<<itt.get_itemname()<<setw(15)
<<itt.get_cost()<<setw(18)<<itt.get_rest()<<endl;


		acf.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));
	}


	acf.close();//κλεινουμε το αρχειο ειδων


	for (r=0;r<77;r++)
		cout<<"-";
	cout<<endl<<endl;
}



void update_item(int cod)
{
	bool found=false;
	double newcost;
	int r,i=0;


	fstream acf("items.txt",ios::in|ios::out);
	ItemData itt;
```

```
cout<<setw(20)<<"UPDATE ITEM"<<endl;
for (r=0;r<20;r++)
        cout<<"-";

cout<<endl;

while(!acf.eof())
{
        acf.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));

        i++;

        if (itt.get_itemcode()==cod)
        {
                found=true;
                break;
        }
}

if(found==false)
{
        cout<<"Item Code "<<cod<<" does not exist"<<endl;
        return;
}

cout<<endl<<"Item exists and is the following one:"<<endl;
cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"ITEMNAME"<<setw(15)<<"
COST/ITEM"<<setw(15)<<"REST"<<endl;
cout<<left<<setw(7)<<itt.get_itemcode()<<setw(13)<<itt.get_itemname()<<setw(15)
<<itt.get_cost()<<setw(18)<<itt.get_rest()<<endl;
```

```
	cout<<"Update Item Cost"<<endl;

	cout<<"Enter new Item's cost "<<endl;
	cin>>newcost;

	itt.set_cost(newcost);

	acf.seekg((i-1)*sizeof(ItemData));//H συναρτηση seekg Μεταφερει το δεικτη του αρχειου σε μια συγκεκριμενη θεση.Εδω μεταφερουμε
                                                                //το δεικτη του αρχειου στη προηγουμενη εγγραφη αυτης που θελουμε να ενημεροσουμε το υπολοιπο

	acf.write(reinterpret_cast<char*>(&itt),sizeof(ItemData));

	cout<<"Item cost succesfully changed!"<<endl;


	acf.close();//κλεινουμε το αρχειο πελατών
}


void update_items_rest(int cod,int q)
{
	bool found=false;
	int i=0;

	fstream acf("items.txt",ios::in|ios::out);
	ItemData itt;

	while(!acf.eof())
```

```
	{
		acf.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));

		i++;

		if (itt.get_itemcode()==cod)
		{
			found=true;
			break;
		}
	}

	itt.set_rest(itt.get_rest()-q);

	cout<<endl<<"Item with new rest after new buying of  quantity "<<q<<" is:"<<endl;
	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"ITEMNAME"<<setw(15)<<"
COST/ITEM"<<setw(15)<<"REST"<<endl;
	cout<<left<<setw(7)<<itt.get_itemcode()<<setw(13)<<itt.get_itemname()<<setw(15)
<<itt.get_cost()<<setw(18)<<itt.get_rest()<<endl;


	acf.seekg((i-1)*sizeof(ItemData));//Η συναρτηση seekg Μεταφερει το δεικτη του
αρχειου σε μια συγκεκριμενη θεση.Εδω μεταφερουμε
                                                    //το δεικτη του
αρχειου στη προηγουμενη εγγραφη αυτης που θελουμε να ενημεροσουμε το υπολοιπο

	acf.write(reinterpret_cast<char*>(&itt),sizeof(ItemData));

	acf.close();//κλεινουμε το αρχειο πελατών
}
```

```
void update_items_rest_buing(int cod,int q)
{
	bool found=false;
	int i=0;

	fstream acf("items.txt",ios::in|ios::out);
	ItemData itt;

	while(!acf.eof())
	{
		acf.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));

		i++;

		if (itt.get_itemcode()==cod)
		{
			found=true;
			break;
		}
	}

	itt.set_rest(itt.get_rest()+q);

	cout<<endl<<"Item with new rest after new buying of  quantity "<<q<<" is:"<<endl;
	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"ITEMNAME"<<setw(15)<<"
COST/ITEM"<<setw(15)<<"REST"<<endl;
	cout<<left<<setw(7)<<itt.get_itemcode()<<setw(13)<<itt.get_itemname()<<setw(15)
<<itt.get_cost()<<setw(18)<<itt.get_rest()<<endl;
```

```
	acf.seekg((i-1)*sizeof(ItemData));//Η συναρτηση seekg Μεταφερει το δεικτη του αρχειου σε μια συγκεκριμενη θεση.Εδω μεταφερουμε
                                                                  //το δεικτη του αρχειου στη προηγουμενη εγγραφη αυτης που θελουμε να ενημεροσουμε το υπολοιπο

	acf.write(reinterpret_cast<char*>(&itt),sizeof(ItemData));

	acf.close();//κλεινουμε το αρχειο πελατών
}

void delete_item(int cod)
{
	bool found=false;
	char ans;
	int r,i=0,j=0,n,pos;

	fstream acf("items.txt",ios::in|ios::out),acf3("items.txt",ios::in|ios::out);
	ItemData itt;

	cout<<setw(20)<<"DELETE ITEM"<<endl;
	for (r=0;r<20;r++)
		cout<<"-";

	cout<<endl;


	while(!acf.eof())
	{
		acf.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));

		i++;
```

```
	}

	n=i-1;//Υπολογίζουμε το πλήθος των εγγραφών του αρχείου
	ItemData *cd=new ItemData[n];//Δεσμευμε δυναμηκα μνημη για να τις τοποθετησουμε σε πίνακα

	while(!acf3.eof())
		acf3.read(reinterpret_cast<char*>(&cd[j++]),sizeof(ItemData));//διαβαζουμε μια εγγραφή πελάτη και τη βάζοθμε στο πίνακα πελατών

	for (i=0;i<n;i++)
	{
		if (cd[i].get_itemcode()==cod)
		{
			pos=i;//Αν το είδος υπάρχει θέτουμε στη μεταβλητή found την τιμή true
			found=true;
			break;
		}
	}

	if(found==false)
	{
		cout<<"Item Code "<<cod<<" does not exist"<<endl;
		return;
	}

	cout<<endl<<"Item exists and is the following one:"<<endl;
	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"ITEMNAME"<<setw(15)<<" COST/ITEM"<<setw(15)<<"REST"<<endl;
```

```
	cout<<left<<setw(7)<<cd[pos].get_itemcode()<<setw(13)<<cd[pos].get_itemname()
<<setw(15)<<cd[pos].get_cost()<<setw(18)<<cd[pos].get_rest()<<endl;


	cout<<"DO YOU WANT TO DELETE THIS? (Y/N)"<<endl;
	cin>>ans;


	if (ans=='y' || ans=='Y')
	{


		for (i=pos;i<n-1;i++)//Κάνουμε αριστερή ολισθιση κατα μια θεση στους
πελάτες
			cd[i]=cd[i+1];


		n--;//Μειωνουμε το μεγεθος των πελατών κατα 1


		fstream acf2("items.txt",ios::out);


		for(i=0;i<n;i++)//
			acf2.write(reinterpret_cast<char*>(&cd[i]),sizeof(ItemData));


		acf2.close();//κλεινουμε το αρχειο πελατών


		cout<<"Item succesfully deleted!"<<endl;
	}


	acf.close();//κλεινουμε το αρχειο πελατών
}



void print_items_rest(int cod)
{
```

```
ifstream acf("items.txt");
ItemData itt;
int r;
bool found=false;

if (acf.fail()==true)
{
	cout<<"Item's File does not exists!"<<endl;
	ofstream acf("items.txt");
	return;
}

cout<<setw(39)<<"ITEMS REST"<<endl;
for (r=0;r<43;r++)
	cout<<"-";

cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"ITEMNAME"<<setw(15)<<"
COST/ITEM"<<setw(15)<<"REST"<<endl;

acf.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));

while(!acf.eof())
{
	if (itt.get_itemcode()==cod)
	{

cout<<left<<setw(7)<<itt.get_itemcode()<<setw(13)<<itt.get_itemname()<<setw(15)
<<itt.get_cost()<<setw(18)<<itt.get_rest()<<endl;
		found=true;
		break;
	}
```

```
            acf.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));
        }

        acf.close();//κλεινουμε το αρχειο προμηθευτων

        for (r=0;r<43;r++)
            cout<<"-";
        cout<<endl<<endl;

        if (found==false)
            cout<<"Item does not exists"<<endl;
}


void print_items_in_lack(int limit)
{
        ifstream acf("items.txt");
        ItemData itt;
        int r;

        if (acf.fail()==true)
        {
            cout<<"Item's File does not exists!"<<endl;
            ofstream acf("items.txt");
            return;
        }

        cout<<setw(39)<<"ITEMS IN LACK"<<endl;
        for (r=0;r<43;r++)
```

```
		cout<<"-";

	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"ITEMNAME"<<setw(15)<<"
COST/ITEM"<<setw(15)<<"REST"<<endl;

	acf.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));

	while(!acf.eof())
	{
		if (itt.get_rest()<limit)

	cout<<left<<setw(7)<<itt.get_itemcode()<<setw(13)<<itt.get_itemname()<<setw(15)
<<itt.get_cost()<<setw(18)<<itt.get_rest()<<endl;

		acf.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));
	}

	acf.close();//κλεινουμε το αρχειο προμηθευτων

	for (r=0;r<43;r++)
		cout<<"-";

	cout<<endl<<endl;
}

void print_item_sales(int itemcode)
{
	ifstream acf("salecustomer.txt");
	SaleCustomerData salecust;
	int r;
```

```
if (acf.fail()==true)
{
	cout<<"Customer's Sales File does not exists!"<<endl;
	ofstream acf("salecustomer.txt");
	return;
}

cout<<setw(39)<<"CUSTOMER SALES"<<endl;
for (r=0;r<77;r++)
	cout<<"-";

cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(9)<<"ITEM"<<setw(11)<<"QUANTITY"<<setw(18)<<"AMOUNT"<
<setw(8)<<"DATE"<<endl;

acf.read(reinterpret_cast<char*>(&salecust),sizeof(SaleCustomerData));

while(!acf.eof())
{
	if (salecust.get_itemcode()==itemcode)

cout<<left<<setw(7)<<salecust.get_codecustomer()<<setw(13)<<salecust.get_lastna
me()<<setw(15)<<salecust.get_firstname()<<setw(9)<<salecust.get_itemcode()<<setw(11)<
<salecust.get_quantity()<<right<<setw(4)<<salecust.get_amount()<<setw(7)<<salecust.get_
day()<<"/"<<salecust.get_month()<<"/"<<salecust.get_year()<<endl;

	acf.read(reinterpret_cast<char*>(&salecust),sizeof(SaleCustomerData));
}

acf.close();//κλεινουμε το αρχειο πωλήσεων

for (r=0;r<77;r++)
```

```
		cout<<"-";
	cout<<endl<<endl;
}

void print_item_purchases(int itemcode)
{
	ifstream acf("salesupplier.txt");
	SaleSupplierData salesupp;
	int r;

	if (acf.fail()==true)
	{
		cout<<"Supplier's Purchase File does not exists!"<<endl;
		ofstream acf("salesupplier.txt");
		return;
	}

	cout<<setw(39)<<"SUPPLIER PURCHASES"<<endl;
	for (r=0;r<77;r++)
		cout<<"-";

	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(9)<<"ITEM"<<setw(11)<<"QUANTITY"<<setw(18)<<"AMOUNT"<
<setw(8)<<"DATE"<<endl;

	acf.read(reinterpret_cast<char*>(&salesupp),sizeof(SaleSupplierData));

	while(!acf.eof())
	{
		if (salesupp.get_itemcode()==itemcode)
```

```
	cout<<left<<setw(7)<<salesupp.get_codesupplier()<<setw(13)<<salesupp.get_lastna
me()<<setw(15)<<salesupp.get_firstname()<<setw(9)<<salesupp.get_itemcode()<<setw(11)
<<salesupp.get_quantity()<<right<<setw(4)<<salesupp.get_amount()<<setw(7)<<salesupp.g
et_day()<<"/"<<salesupp.get_month()<<"/"<<salesupp.get_year()<<endl;


			acf.read(reinterpret_cast<char*>(&salesupp),sizeof(SaleSupplierData));
	}


	acf.close();//κλεινουμε το αρχειο πωλήσεων


	for (r=0;r<77;r++)
		cout<<"-";
	cout<<endl<<endl;
}
```

3.13 Paymentcustomer.cpp

```
#include <iostream>

using std::string;

#include <cstring>


#include "epixeir.h"//αυτό το hader file δεν βρίσκεται στο φάκελο που είναι όλα τα header files αλλά στον τρέχοντα φάκελο. Γιαυτό το όνομα του μπαίνει σε ""

PaymentCustomerData::PaymentCustomerData(int cn,char l[],char f[],double am,int day,int month,int year)

{

	codecustomer=cn;

	strcpy(lastname,l);

	strcpy(firstname,f);

	amount=am;

	date.day=day;

	date.month=month;

	date.year=year;

}


void PaymentCustomerData::set_codecustomer(int cn)

{

	codecustomer=cn;

}


void PaymentCustomerData::set_lastname(char l[])

{

	strcpy(lastname,l);

}


void PaymentCustomerData::set_firstname(char f[])
```

```
{
        strcpy(firstname,f);
}

void PaymentCustomerData::set_amount(double am)
{
        amount=am;
}

void PaymentCustomerData::set_date(int day,int month,int year)
{
        date.day=day;

        date.month=month;

        date.year=year;
}

int PaymentCustomerData::get_codecustomer()
{
        return codecustomer;
}

char* PaymentCustomerData::get_lastname()
{
        return lastname;
}

char* PaymentCustomerData::get_firstname()
{
```

```
	return firstname;
}

double PaymentCustomerData::get_amount()
{
	return amount;
}

int PaymentCustomerData::get_day()
{
	return date.day;
}

int PaymentCustomerData::get_month()
{
	return date.month;
}

int PaymentCustomerData::get_year()
{
	return date.year;
}
```

3.14 Paymentsupplier.cpp

```
#include <iostream>

using std::string;


#include <cstring>


#include "epixeir.h"//αυτό το hader file δεν βρίσκεται στο φάκελο που είναι όλα τα header files αλλά στον τρέχοντα φάκελο. Γιαυτό το όνομα του μπαίνει σε ""


PaymentSupplierData::PaymentSupplierData(int cn,char l[],char f[],double am,int day,int month,int year)
{
	codesupplier=cn;
	strcpy(lastname,l);
	strcpy(firstname,f);
	amount=am;
	date.day=day;
	date.month=month;
	date.year=year;
}



void PaymentSupplierData::set_codesupplier(int cn)
{
	codesupplier=cn;
}


void PaymentSupplierData::set_lastname(char l[])
{
	strcpy(lastname,l);
```

```
}

void PaymentSupplierData::set_firstname(char f[])
{
	strcpy(firstname,f);
}

void PaymentSupplierData::set_amount(double am)
{
	amount=am;
}

void PaymentSupplierData::set_date(int day,int month,int year)
{
	date.day=day;

	date.month=month;

	date.year=year;
}

int PaymentSupplierData::get_codesupplier()
{
	return codesupplier;
}

char* PaymentSupplierData::get_lastname()
{
	return lastname;
}
```

```
char* PaymentSupplierData::get_firstname()
{
	return firstname;
}

double PaymentSupplierData::get_amount()
{
	return amount;
}

int PaymentSupplierData::get_day()
{
	return date.day;
}

int PaymentSupplierData::get_month()
{
	return date.month;
}

int PaymentSupplierData::get_year()
{
	return date.year;
}
```

3.15 Salecustomer.cpp

```
#include <iostream>

using std::string;


#include <cstring>


#include "epixeir.h"//αυτό το hader file δεν βρίσκεται στο φάκελο που είναι όλα τα header files αλλά στον τρέχοντα φάκελο. Γιαυτό το όνομα του μπαίνει σε ""



SaleCustomerData::SaleCustomerData(int cn,char l[],char f[],double am,int day,int month,int year,int itcode,int itqua)

{

        codecustomer=cn;


        strcpy(lastname,l);


        strcpy(firstname,f);


        amount=am;


        date.day=day;


        date.month=month;


        date.year=year;


        itemcode=itcode;


        quantity=itqua;
```

```
}


void SaleCustomerData::set_codecustomer(int cn)
{
        codecustomer=cn;
}

void SaleCustomerData::set_lastname(char l[])
{
        strcpy(lastname,l);
}

void SaleCustomerData::set_firstname(char f[])
{
        strcpy(firstname,f);
}


void SaleCustomerData::set_amount(double am)
{
        amount=am;
}

void SaleCustomerData::set_date(int day,int month,int year)
{
        date.day=day;

        date.month=month;
```

```
	date.year=year;
}

void SaleCustomerData::set_itemcode(int itcode)
{
	itemcode=itcode;
}

void SaleCustomerData::set_quantity(int itqua)
{
	quantity=itqua;
}

int SaleCustomerData::get_codecustomer()
{
	return codecustomer;
}

char* SaleCustomerData::get_lastname()
{
	return lastname;
}

char* SaleCustomerData::get_firstname()
{
	return firstname;
}


double SaleCustomerData::get_amount()
```

```
{
        return amount;
}

int SaleCustomerData::get_day()
{
        return date.day;
}

int SaleCustomerData::get_month()
{
        return date.month;
}

int SaleCustomerData::get_year()
{
        return date.year;
}

int SaleCustomerData::get_itemcode()
{
        return itemcode;
}

int SaleCustomerData::get_quantity()
{
        return quantity;
}
```

3.16 Salesupplier.cpp

```
#include <iostream>

using std::string;


#include <cstring>


#include "epixeir.h"//αυτό το hader file δεν βρίσκεται στο φάκελο που είναι όλα τα header files αλλά στον τρέχοντα φάκελο. Γιαυτό το όνομα του μπαίνει σε ""



SaleSupplierData::SaleSupplierData(int cn,char l[],char f[],double am,int day,int month,int year,int itcode,int itqua)

{

        codesupplier=cn;


        strcpy(lastname,l);


        strcpy(firstname,f);


        amount=am;


        date.day=day;


        date.month=month;


        date.year=year;


        itemcode=itcode;


        quantity=itqua;
```

```
}


void SaleSupplierData::set_codesupplier(int cn)
{
        codesupplier=cn;
}

void SaleSupplierData::set_lastname(char l[])
{
        strcpy(lastname,l);
}

void SaleSupplierData::set_firstname(char f[])
{
        strcpy(firstname,f);
}


void SaleSupplierData::set_amount(double am)
{
        amount=am;
}

void SaleSupplierData::set_date(int day,int month,int year)
{
        date.day=day;

        date.month=month;
```

```
	date.year=year;
}

void SaleSupplierData::set_itemcode(int itcode)
{
	itemcode=itcode;
}

void SaleSupplierData::set_quantity(int itqua)
{
	quantity=itqua;
}



int SaleSupplierData::get_codesupplier()
{
	return codesupplier;
}

char* SaleSupplierData::get_lastname()
{
	return lastname;
}

char* SaleSupplierData::get_firstname()
{
	return firstname;
}
```

```
double SaleSupplierData::get_amount()
{
	return amount;
}

int SaleSupplierData::get_day()
{
	return date.day;
}

int SaleSupplierData::get_month()
{
	return date.month;
}

int SaleSupplierData::get_year()
{
	return date.year;
}

int SaleSupplierData::get_itemcode()
{
	return itemcode;
}
int SaleSupplierData::get_quantity()
{
	return quantity;
}
```

3.17 Suppliers.cpp

```
#include <iostream>

using std::string;


#include <cstring>


#include "epixeir.h"//αυτό το hader file δεν βρίσκεται στο φάκελο που είναι όλα τα header files αλλά στον τρέχοντα φάκελο. Γιαυτό το όνομα του μπαίνει σε ""



SupplierData::SupplierData(int c,char l[],char f[],char a[],char t[],double b)

{

	codesupplier=c;


	strcpy(lastname,l);


	strcpy(firstname,f);


	strcpy(adresscode,a);


	strcpy(telephonenumber,t);


	balance=b;

}



void SupplierData::set_codesupplier(int c)

{

	codesupplier=c;
```

```
}

void SupplierData::set_lastname(char l[])
{
	strcpy(lastname,l);
}

void SupplierData::set_firstname(char f[])
{
	strcpy(firstname,f);
}

void SupplierData::set_adresscode(char a[])
{
	strcpy(adresscode,a);
}

void SupplierData::set_telephonenumber(char t[])
{
	strcpy(telephonenumber,t);
}

void SupplierData::set_balance(double b)
{
	balance=b;
}


int SupplierData::get_codesupplier()
{
```

```
	return codesupplier;
}

char* SupplierData::get_lastname()
{
	return lastname;
}

char* SupplierData::get_firstname()
{
	return firstname;
}

char* SupplierData::get_adresscode()
{
	return adresscode;
}
char* SupplierData::get_telephonenumber()
{
	return telephonenumber;
}

double SupplierData::get_balance()
{
	return balance;
}
```

3.18 supplier_functions.cpp

```
#include <iostream>//βιβλιοθηκη με συναρτησεις I/O

using std::cin;

using std::cout;

using std::endl;

using std::ios;

using std::cerr;

using std::left;

using std::right;

using std::fixed;


#include <fstream>//βιβλιοθηκη με συναρτησεις διαχειρισεις αρχειων

using std::ofstream;//τυπος για καταχωριση σε αρχειο (αρχεία εξόδου)

using std::ifstream;//τυπος για διάβασμα απο αρχειο (αρχεία εισόδου)

using std::fstream;//τυπος για διάβασμα απο αρχειο και εκτυπωση σε αρχείο (αρχεία εισόδου-εξόδου)


#include <cstdlib>


#include <iomanip>//περιεχει τις συναρτησεις setw ,setprecision.

using std::setw;

using std::setprecision;


void
cls(),insert_salesupplier(),insert_supplier(),delete_suppliers(),insert_supplier(),print_suppliers
(),update_supplier(int),delete_supplier(int),update_supplier_balance(int,double),update_items
_rest_buing(int,int),
print_client_sales(int),print_supplier_sales(int),print_supplier_payments(int),insert_payments
upplier(),print_suppliers_balance(int),total_average_sales_per_supplier();


#include "epixeir.h"
```

```
void delete_suppliers()
{
	char ans[10];

	cout<<"Attention!!"<<endl;
	cout<<"Really delete file of suppliers y/n"<<endl;
	cin>>ans;

	if (strcmp(ans,"y")==0 || strcmp(ans,"Y")==0)
	{
		ofstream acf("supplier.txt");
		ofstream acf1("paymentsupplier.txt");
		ofstream acf2("salesupplier.txt");
	}

}

void insert_supplier()
{
	int r,sup;
	char fn[20],ln[30],adrc[30],teln[10];
	double bal;

	ofstream acf("supplier.txt",ios::app);
	SupplierData supplier;

	cout<<setw(20)<<"INSERT NEW SUPPLIER"<<endl;
	for (r=0;r<20;r++)
		cout<<"-";
```

```
        cout<<endl<<"Enter Supplier's Code: "<<endl;
        cin>>sup;
        supplier.set_codesupplier(sup);

        cout<<"Enter Supplier's First Name: "<<endl;
        cin>>fn;
        supplier.set_firstname(fn);

        cout<<"Enter Supplier's Last Name: "<<endl;
        cin>>ln;
        supplier.set_lastname(ln);

        cout<<"Enter Supplier's Address Code: "<<endl;
        cin>>adrc;

        supplier.set_adresscode(adrc);

        cout<<"Enter Supplier's Telephone Number: "<<endl;
        cin>>teln;
        supplier.set_telephonenumber(teln);

        cout<<"Enter Supplier's Initial Balance: "<<endl;
        cin>>bal;
        supplier.set_balance(bal);

        acf.write(reinterpret_cast<const char*>(&supplier),sizeof(ClientData));

        acf.close();//κλεινουμε το αρχειο προμηθευτων
}
```

```
void insert_paymentsupplier()
{
	int r,cn,day,month,year;
	double amount;
	bool found=false;


	ofstream acf("paymentsupplier.txt",ios::app);
	PaymentSupplierData paysupp;

	if (acf.fail()==true)
	{
		cout<<"Payment Supplier File does not exists!"<<endl;
		ofstream acf("paymentsupplier.txt");
		return;
	}

	cout<<setw(20)<<"INSERT NEW SUPPLIER PAYMENT "<<endl;
	for (r=0;r<20;r++)
		cout<<"-";

	cout<<endl<<"Enter The Supplier's Code: "<<endl;
	cin>>cn;

	ifstream acfc("supplier.txt",ios::in);
	SupplierData supp;

	acfc.read(reinterpret_cast<char*>(&supp),sizeof(SupplierData));
```

```
while(!acfc.eof())
{
	if (supp.get_codesupplier()!=0)
		if (cn==supp.get_codesupplier())
		{
			paysupp.set_codesupplier(cn);
			paysupp.set_firstname(supp.get_firstname());
			paysupp.set_lastname(supp.get_lastname());
			found=true;
		}

	acfc.read(reinterpret_cast<char*>(&supp),sizeof(SupplierData));
}

acfc.close();//κλεινουμε το αρχειο πελατων

if (found==false)
{
	cout<<"Supplier doesn't exists!"<<endl;
	return;
}

cout<<"Enter The Day/Month/Year of Payment"<<endl;
do
{
	cin>>day;

	if (day<1 || day>31)
		cout<<"Wrong Day. Enter another day from 1-31"<<endl;
```

```
}
while (day<1 || day>31);

do
{
        cin>>month;

        if (month<1 || month>12)
                cout<<"Wrong Month. Enter another month from 1-12"<<endl;

}
while(month<1 ||month>12);

do
{
        cin>>year;

        if (year<1998 || year>3000)
                cout<<"Wrong Year. Enter another year from 1998-2100"<<endl;

}
while(year<1998 || year>2100);

cout<<"Enter payments Amount"<<endl;
cin>>amount;

paysupp.set_amount(amount);
paysupp.set_date(day,month,year);

acf.write(reinterpret_cast<const char*>(&paysupp),sizeof(PaymentSupplierData));
```

```
	acf.close();//κλεινουμε το αρχειο πωλήσεων

	update_supplier_balance(cn,-amount);
}


void print_supplier_payments(int sup)
{
	ifstream acf("paymentsupplier.txt");
	PaymentSupplierData paysupp;
	int r;

	if (acf.fail()==true)
	{
		cout<<"Supplier's Payments File does not exists!"<<endl;
		ofstream acf("paymentsupplier.txt");
		return;
	}

	cout<<setw(39)<<"SUPPLIER PAYMENTS"<<endl;
	for (r=0;r<77;r++)
		cout<<"-";

	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(9)<<"AMOUNT"<<right<<setw(12)<<"DATE"<<endl;

	acf.read(reinterpret_cast<char*>(&paysupp),sizeof(PaymentSupplierData));

	while(!acf.eof())
	{
```

```
                if (paysupp.get_codesupplier()==sup)

        cout<<left<<setw(7)<<paysupp.get_codesupplier()<<setw(13)<<paysupp.get_lastna
me()<<setw(15)<<paysupp.get_firstname()<<setw(9)<<paysupp.get_amount()<<setw(7)<<ri
ght<<paysupp.get_day()<<"/"<<paysupp.get_month()<<"/"<<paysupp.get_year()<<endl;


                acf.read(reinterpret_cast<char*>(&paysupp),sizeof(PaymentSupplierData));
        }


        acf.close();//κλεινουμε το αρχειο πληρωμων


        for (r=0;r<77;r++)
                cout<<"-";
        cout<<endl<<endl;
}


void insert_salesupplier()
{
        int r,cn,day,month,year,pr,q;
        double tm;
   bool found=false;


        ofstream acf("salesupplier.txt",ios::app);
        SaleSupplierData salesupp;


        cout<<setw(20)<<"INSERT NEW SALE SUPPLIER"<<endl;
        for (r=0;r<20;r++)
                cout<<"-";


        cout<<endl<<"Enter The Supplier's Code: "<<endl;
        cin>>cn;
```

```
salesupp.set_codesupplier(cn);

ifstream acfc("supplier.txt",ios::in);
SupplierData supp;

acfc.read(reinterpret_cast<char*>(&supp),sizeof(SupplierData));

while(!acfc.eof())
{
        if (supp.get_codesupplier()!=0)
                if (cn==supp.get_codesupplier())
                {
                        salesupp.set_codesupplier(cn);
                        salesupp.set_firstname(supp.get_firstname());
                        salesupp.set_lastname(supp.get_lastname());
                        found=true;
                }
        acfc.read(reinterpret_cast<char*>(&supp),sizeof(SupplierData));
}

acfc.close();//κλεινουμε το αρχειο προμηθευτων

if (found==false)
{
        cout<<"Supplier doesn't exists!"<<endl;
        return;
}

cout<<"Enter Product Code "<<endl;
cin>>pr;
```

```
ifstream acfp("items.txt",ios::in);

if (acfp.fail()==true)
{
	cout<<"Items File not exists!"<<endl;
	ofstream acf("items.txt");
	return;
}

ItemData itt;

acfp.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));

found=false;

while(!acfp.eof())
{
	if (itt.get_itemcode()!=0)
		if (pr==itt.get_itemcode())
		{
			salesupp.set_itemcode(pr);
			tm=itt.get_cost();
			found=true;
		}

	acfp.read(reinterpret_cast<char*>(&itt),sizeof(ItemData));
}

acfp.close();//κλεινουμε το αρχειο ειδων
```

```
if (found==false)
{
	cout<<"Item doesn't exists!"<<endl;
	return;
}

tm=itt.get_cost();

cout<<"Enter Product Quantity "<<endl;
cin>>q;

salesupp.set_amount(q*tm);


cout<<"Enter The Day/Month/Year Of SALE"<<endl;
do
{
	cin>>day;

	if (day<1 || day>31)
		cout<<"Wrong Day. Enter another day from 1-31"<<endl;
}
while (day<1 || day>31);

do
{
	cin>>month;

	if (month<1 || month>12)
```

```
                cout<<"Wrong Month. Enter another month from 1-12"<<endl;

        }
        while(month<1 ||month>12);

        do
        {
                cin>>year;

                if (year<1998 || year>3000)
                        cout<<"Wrong Year. Enter another year from 1998-2100"<<endl;

        }
        while(year<1998 || year>2100);

        salesupp.set_quantity(q);
        salesupp.set_amount(q*tm);
        salesupp.set_date(day,month,year);


        acf.write(reinterpret_cast<const char*>(&salesupp),sizeof(SaleSupplierData));

        acf.close();//κλεινουμε το αρχειο προμηθευτων

        update_supplier_balance(cn,q*tm);

        update_items_rest_buing(pr,q);

}
```

```
void print_suppliers()
{
	ifstream acf("supplier.txt",ios::in);

	if (acf.fail()==true)
	{
		cout<<"Suppliers File not exists!"<<endl;
		ofstream acf("supplier.txt");
		return;
	}

	SupplierData supplier;
	int r;

	cout<<setw(39)<<"SUPPLIERS"<<endl;
	for (r=0;r<77;r++)
		cout<<"-";
	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(18)<<"ADDRESS"<<setw(17)<<"TELEPHONE"<<setw(15)<<"BAL
ANCE"<<endl;

	acf.read(reinterpret_cast<char*>(&supplier),sizeof(SupplierData));

	while(!acf.eof())
	{
		if (supplier.get_codesupplier()!=0)

	cout<<left<<setw(7)<<supplier.get_codesupplier()<<setw(13)<<supplier.get_lastnam
e()<<setw(15)<<supplier.get_firstname()<<setw(18)<<supplier.get_adresscode()<<setw(17)
<<supplier.get_telephonenumber()<<right<<setw(7)<<supplier.get_balance()<<endl;

		acf.read(reinterpret_cast<char*>(&supplier),sizeof(SupplierData));
```

```
	}

	acf.close();//κλεινουμε το αρχειο προμηθευτων

	for (r=0;r<77;r++)
		cout<<"-";
	cout<<endl<<endl;
}

void print_supplier_sales(int supp)
{
	ifstream acf("salesupplier.txt");
	SaleSupplierData salesupp;
	int r;

	if (acf.fail()==true)
	{
		cout<<"Supplier's Purchase File does not exists!"<<endl;
		ofstream acf("salesupplier.txt");
		return;
	}

	cout<<setw(39)<<"SUPPLIER PURCHASES"<<endl;
	for (r=0;r<77;r++)
		cout<<"-";

	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(9)<<"ITEM"<<setw(11)<<"QUANTITY"<<setw(18)<<"AMOUNT"<
<setw(8)<<"DATE"<<endl;

	acf.read(reinterpret_cast<char*>(&salesupp),sizeof(SaleSupplierData));
```

```
	while(!acf.eof())
	{
		if (salesupp.get_codesupplier()==supp)

	cout<<left<<setw(7)<<salesupp.get_codesupplier()<<setw(13)<<salesupp.get_lastna
me()<<setw(15)<<salesupp.get_firstname()<<setw(9)<<salesupp.get_itemcode()<<setw(11)
<<salesupp.get_quantity()<<right<<setw(4)<<salesupp.get_amount()<<setw(7)<<salesupp.g
et_day()<<"/"<<salesupp.get_month()<<"/"<<salesupp.get_year()<<endl;


		acf.read(reinterpret_cast<char*>(&salesupp),sizeof(SaleSupplierData));
	}


	acf.close();//κλεινουμε το αρχειο πωλήσεων


	for (r=0;r<77;r++)
		cout<<"-";
	cout<<endl<<endl;
}



void print_suppliers_balance(int suppbal)
{
	ifstream acf("supplier.txt");
	SupplierData supp;
	int r;
	bool found=false;

	if (acf.fail()==true)
	{
		cout<<"Supplier's File does not exists!"<<endl;
		ofstream acf("supplier.txt");
```

```
            return;
        }

        cout<<setw(39)<<"SUPPLIER BALANCE"<<endl;
        for (r=0;r<43;r++)
            cout<<"-";

        cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(18)<<setw(15)<<"BALANCE"<<endl;

        acf.read(reinterpret_cast<char*>(&supp),sizeof(SupplierData));

        while(!acf.eof())
        {
            if (supp.get_codesupplier()==suppbal)
            {

        cout<<left<<setw(7)<<supp.get_codesupplier()<<setw(13)<<supp.get_lastname()<<s
etw(15)<<supp.get_firstname()<<setw(18)<<right<<setw(7)<<supp.get_balance()<<endl;
                found=true;
                break;
            }

            acf.read(reinterpret_cast<char*>(&supp),sizeof(SupplierData));
        }

        acf.close();//κλεινουμε το αρχειο προμηθευτων

        for (r=0;r<43;r++)
            cout<<"-";
        cout<<endl<<endl;
```

```
	if (found==false)
		cout<<"Supplier does not exists"<<endl;

}


void update_supplier(int supp)
{
	bool found=false;
	char type;
	char newaddr[30],newtel[10];
	int r,i=0;

	fstream acf("supplier.txt",ios::in|ios::out);
	SupplierData supplier;

	cout<<setw(20)<<"UPDATE SUPPLIER"<<endl;
	for (r=0;r<20;r++)
		cout<<"-";

	while(!acf.eof())
	{
		acf.read(reinterpret_cast<char*>(&supplier),sizeof(SupplierData));

		i++;

		if (supplier.get_codesupplier()==supp)
		{
			found=true;
```

```
                break;
            }
        }

        if(found==false)
        {
            cout<<"Supplier Code "<<supp<<" does not exist"<<endl;
            return;
        }

        cout<<endl<<"Supplier exists and is the following one:"<<endl;
        cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(18)<<"ADDRESS"<<setw(17)<<"TELEPHONE"<<setw(15)<<"BAL
ANCE"<<endl;
        cout<<left<<setw(7)<<supplier.get_codesupplier()<<setw(13)<<supplier.get_lastnam
e()<<setw(15)<<supplier.get_firstname()<<setw(18)<<supplier.get_adresscode()<<setw(17)
<<supplier.get_telephonenumber()<<right<<setw(7)<<supplier.get_balance()<<endl;



        cout<<"Update Supplier's (A)ddress or (T)elephone?"<<endl;
        cin>>type;

        if (type=='a' || type=='A')
        {
            cout<<"Enter new Supplier's Address"<<endl;
            cin>>newaddr;

            supplier.set_adresscode(newaddr);

            cout<<"Supplier's address succesfully changed!"<<endl;

        }
```

```
	if (type=='t' || type=='T')
	{
		cout<<"Enter new Supplier's Telephone"<<endl;
		cin>>newtel;

		supplier.set_telephonenumber(newtel);

		cout<<"Supplier's  telephone succesfully changed!"<<endl;

	}


	acf.seekg((i-1)*sizeof(SupplierData));//Η συναρτηση seekg Μεταφερει το δεικτη του αρχειου σε μια συγκεκριμενη θεση.Εδω μεταφερουμε
                                                                  //το δεικτη του αρχειου στη προηγουμενη εγγραφη αυτης που θελουμε να ενημεροσουμε το υπολοιπο

	acf.write(reinterpret_cast<char*>(&supplier),sizeof(SupplierData));

	acf.close();//κλεινουμε το αρχειο προμηθευτων
}


void update_supplier_balance(int cust,double am)
{
	bool found=false;
	int i=0;

	fstream acf("supplier.txt",ios::in|ios::out);
	SupplierData supp;
```

```
	while(!acf.eof())
	{
		acf.read(reinterpret_cast<char*>(&supp),sizeof(SupplierData ));

		i++;

		if (supp.get_codesupplier()==cust)
		{
			found=true;
			supp.set_balance(supp.get_balance()+am);
			break;
		}
	}

	cout<<endl<<"Supplier with new balance after new buying of  amount "<<am<<" is:"<<endl;
	cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"FIRSTNAME"<<setw(18)<<"ADDRESS"<<setw(17)<<"TELEPHONE"<<setw(15)<<"BALANCE"<<endl;
	cout<<left<<setw(7)<<supp.get_codesupplier()<<setw(13)<<supp.get_lastname()<<setw(15)<<supp.get_firstname()<<setw(18)<<supp.get_adresscode()<<setw(17)<<supp.get_telephonenumber()<<right<<setw(7)<<supp.get_balance()<<endl;

	acf.seekg((i-1)*sizeof(SupplierData));//Η συναρτηση seekg Μεταφερει το δεικτη του αρχειου σε μια συγκεκριμενη θεση.Εδω μεταφερουμε
                                              //το δεικτη του αρχειου στη προηγουμενη εγγραφη αυτης που θελουμε να ενημεροσουμε το υπολοιπο

	acf.write(reinterpret_cast<char*>(&supp),sizeof(SupplierData));

	acf.close();//κλεινουμε το αρχειο πωλήσεων
}
```

```
void delete_supplier(int supp)
{
	bool found=false;
	char ans;
	int r,i=0,j=0,n,pos,k;

	fstream acf("supplier.txt",ios::in|ios::out),acf3("supplier.txt",ios::in|ios::out);
	SupplierData supplier;

	cout<<setw(20)<<"DELETE SUPPLIER"<<endl;
	for (r=0;r<20;r++)
		cout<<"-";

	cout<<endl;

	while(!acf.eof())
	{
		acf.read(reinterpret_cast<char*>(&supplier),sizeof(SupplierData));

		i++;
	}

	n=i-1;//Υπολογίζουμε το πλήθος των εγγραφών του αρχείου
	SupplierData *cd=new SupplierData[n];//Δεσμευμε δυναμηκα μνημη για να τις τοποθετησουμε σε πίνακα

	while(!acf3.eof())

	acf3.read(reinterpret_cast<char*>(&cd[j++]),sizeof(SupplierData));//διαβαζουμε μια εγγραφή προμηθευτη και τη βάζοθμε στο πίνακα προμηθευτων
```

```
	for (i=0;i<n;i++)
	{
		if (cd[i].get_codesupplier()==supp)
		{
			pos=i;//Αν ο προμηθευτης υπάρχει βάζουμε  τη θέση του στη μεταβλητή i
			found=true;
			break;
		}
	}

	if(found==false)
	{
		cout<<"Supplier Code "<<supp<<" does not exist"<<endl;
		return;
	}

	cout<<endl<<"Supplier exists and is the following one:"<<endl;
	cout<<left<<setw(12)<<"CODE"<<setw(25)<<"LASTNAME"<<setw(26)<<"FIRST NAME"<<setw(14)<<"BALANCE"<<endl;
	cout<<left<<setw(12)<<cd[pos].get_codesupplier()<<setw(25)<<cd[pos].get_lastname()<<setw(20)<<cd[pos].get_firstname()<<right<<setw(10)<<cd[pos].get_balance()<<endl;

	cout<<"DO YOU WANT TO DELETE HIM/HER? (Y/N)"<<endl;
	cin>>ans;

	if (ans=='y' || ans=='Y')
	{
```

```
		for (i=pos;i<n-1;i++)//Κάνουμε αριστερή ολισθιση κατα μια θεση στους προμηθευτες
			cd[i]=cd[i+1];

		n--;//Μειωνουμε το μεγεθος των προμηθευτων κατα 1

		fstream acf2("supplier.txt",ios::out);

		for(i=0;i<n;i++)
			acf2.write(reinterpret_cast<char*>(&cd[i]),sizeof(SupplierData));

		acf2.close();//κλεινουμε το αρχειο προμηθευτων
		acf.close();//κλεινουμε το αρχειο προμηθευτων

		cout<<"Supplier with code "<<supp<<" has been succesfully deleted!"<<endl;

		ifstream acfs("paymentsupplier.txt",ios::in),acf3s("supplier.txt",ios::in);
		PaymentSupplierData paysupp;

		if (acfs.fail()==true)
		{
			cout<<"Payment Supplier File does not exists!"<<endl;
			ofstream acf("paymentsupplier.txt");
		}

		i=0;

		while(!acfs.eof())
		{
	acfs.read(reinterpret_cast<char*>(&paysupp),sizeof(PaymentSupplierData));
```

```
                i++;
        }

        n=i-1;//Υπολογίζουμε το πλήθος των εγγραφών του αρχείου

        if (n==0)
                cout<<"There aren't any payments to this Supplier"<<endl;
        else
        {
                PaymentSupplierData *cd1=new
PaymentSupplierData[n];//Δεσμεουμε δυναμηκα μνημη για να τις τοποθετησουμε σε πίνακα
                PaymentSupplierData *newcd1=new
PaymentSupplierData[n];//Δεσμεουμε δυναμηκα μνημη για να τις τοποθετησουμε σε πίνακα

                j=0;

                while(!acf3s.eof())

acf3s.read(reinterpret_cast<char*>(&cd1[j++]),sizeof(PaymentSupplierData));

                k=0;

                for (i=0;i<n;i++)
                        if (cd1[i].get_codesupplier()!=supp)
                                newcd1[k++]=cd1[i];


                fstream acf2s("paymentsupplier.txt",ios::out);

                for(i=0;i<k;i++)
```

```
acf2s.write(reinterpret_cast<char*>(&newcd1[i]),sizeof(PaymentSupplierData));

                acf2s.close();
                acfs.close();
                acf3s.close();

                cout<<"All Supplier Payments have been succesfully deleted!"<<endl;
        }

        ifstream acfsc("salesupplier.txt",ios::in),acf3sc("salesupplier.txt",ios::in);
        SaleSupplierData salesupp;

        i=0;

        while(!acfsc.eof())
        {

acfsc.read(reinterpret_cast<char*>(&salesupp),sizeof(SaleSupplierData));

                i++;
        }

        n=i-1;//Υπολογίζουμε το πλήθος των εγγραφών του αρχείου

        if (n==0)
                cout<<"There aren't any purchases to this Supplier"<<endl;
        else
        {
                SaleSupplierData *cd2=new SaleSupplierData[n];//Δεσμευουμε
δυναμηκα μνημη για να τις τοποθετησουμε σε πίνακα
```

```
                    SaleSupplierData *newcd2=new SaleSupplierData[n];//Δεσμευουμε
δυναμηκα μνημη για να τις τοποθετησουμε σε πίνακα

                    j=0;

                    while(!acf3sc.eof())

acf3sc.read(reinterpret_cast<char*>(&cd2[j++]),sizeof(SaleSupplierData));

                    k=0;

                    for (i=0;i<n;i++)
                        if (cd2[i].get_codesupplier()!=supp)
                            newcd2[k++]=cd2[i];

                    fstream acf2sc("salesupplier.txt",ios::out);

                    for(i=0;i<k;i++)

        acf2sc.write(reinterpret_cast<char*>(&newcd2[i]),sizeof(SaleSupplierData));

                    acf2sc.close();
                    acf3sc.close();
                    acfsc.close();

                    cout<<"All Supplier Purchases have been succesfully deleted!"<<endl;
                }

        }

}
```

```
void total_average_sales_per_supplier()
{
	bool found=false;
	int i=0,j=0,n,pol=0;
	double sum=0;

	fstream acf1("supplier.txt",ios::in),acf2("supplier.txt",ios::in);
	SupplierData supp;

	if (acf1.fail()==true)
	{
		cout<<"there are no any supplier!"<<endl;
		ofstream acf("supplier.txt");
	}

	while(!acf1.eof())
	{
		acf1.read(reinterpret_cast<char*>(&supp),sizeof(SupplierData));

		i++;
	}
	acf1.close();

	n=i-1;//Υπολογίζουμε το πλήθος των εγγραφών του αρχείου
	SupplierData *cd=new SupplierData[n];//Δεσμευμε δυναμικα μνημη για να τις τοποθετησουμε σε πίνακα

	while(!acf2.eof())
```

```
        acf2.read(reinterpret_cast<char*>(&cd[j++]),sizeof(SupplierData));//διαβαζουμε μια εγγραφή πελάτη και τη βάζοθμε στο πίνακα πελατών

        SaleSupplierData salesupp;

        cout<<setw(39)<<"TOTAL & AVERAGE  SALES PER SUPPLIER"<<endl;

        for (i=0;i<n;i++)
        {
                ifstream acf3("salesupplier.txt");

                if (acf1.fail()==true)
                {
                        cout<<"there are no any sales from supplier!"<<endl;
                        ofstream acf("salesupplier.txt");
                }

                sum=0;
                pol=0;

                acf3.read(reinterpret_cast<char*>(&salesupp),sizeof(SaleSupplierData));

                while(!acf3.eof())
                {
                        if (salesupp.get_codesupplier()==cd[i].get_codesupplier())
                        {
                                sum+=salesupp.get_amount();
                                pol++;
                        }
```

```
        acf3.read(reinterpret_cast<char*>(&salesupp),sizeof(SaleSupplierData));
                }


        cout<<endl<<left<<setw(7)<<"CODE"<<setw(13)<<"LASTNAME"<<setw(15)<<"F
IRSTNAME"<<setw(9)<<"SALES#"<<setw(11)<<"TOTAL AMOUNT"<<setw(20)<<"
AVG AMOUNT/SALE"<<endl;


        cout<<setw(7)<<cd[i].get_codesupplier()<<setw(13)<<cd[i].get_lastname()<<setw(1
5)<<cd[i].get_firstname()<<setw(11)<<pol<<setw(11)<<sum<<setw(20)<<sum/pol<<endl;
                cout<<endl;
                acf3.close();//κλεινουμε το αρχειο πωλήσεων
        }
}
```

4 Βιβλιογραφία

Βιβλία

Εισαγωγή στη C++
Συγγραφέας: Γιάννης Τσιομπίκας, Κωνσταντίνος Μαργαρίτης.. Συντάκτης: eBooks4Greeks

Η γλώσσα C σε βάθος
Εκδόσεις: Κλειδάριθμος
Συγγραφέας: Νίκος Μ. Χατζηγιαννάκης

Η γλώσσα C++ σε βάθος
Εκδόσεις: Κλειδάριθμος
Συγγραφέας: Νίκος Μ. Χατζηγιαννάκης

Η γλώσσα προγραμματισμού C
Εκδόσεις: Κλειδάριθμος
Συγγραφέας: Brian W. Kernighan

Java προγραμματισμός
Εκδόσεις: Γκιούρδας Μ. (2010)
Συγγραφέας: Paul J. Deitel

Ασκήσεις - προγράμματα σε C++
Εκδόσεις: Γκιούρδας Μ.
Συγγραφέας: Harvey M. Deitel

Μάθετε τη Java 24 ώρες
Συγγραφέας: ΚΑΝΤΕΝΧΕΝΤ ΡΟΤΖΕΡΣ
Εκδοτικός οίκος: ΓΚΙΟΥΡΔΑΣ Μ.

On line Tutorials

C++ Tutorial, C++ Made Easy: Learning to Program in C++

C Tutorial - C Made Easy

More Advanced C and C++ Language Feature Tutorials

Ηλεκτρονικό εγχειρίδιο της JAVA
Συντάκτης: eBooks4Greeks, Συγγραφέας: Παπαδοπούλου Μαρία

5 Παράρτημα – Πηγαίος Κώδικας

5.1 transaction.h

```
#ifndef TRANSACTION_H

#define TRANSACTION _H

class Transaction //κίνηση πελάτη ή προμηθευτή

{

        int code; //Κωδικός πελάτη ή προμηθευτή

        double amount;

        char type[20];

        char date[10];

};
```