Τμήμα Μηχανικών Η/Υ & Πληροφορικής
Πανεπιστήμιο Πατρών

Εισαγωγή στον Προγραμματισμό Υπολογιστών

Η γλώσσα προγραμματισμού C

Δ3 – Τέσσερις Πράξεις (Menu)

Αναπτύξτε ένα πρόγραμμα με όνομα Menu σύμφωνα με το οποίο το σύστημα θα εκτελεί τις υπολογιστικές διεργασίες που περιγράφει η Άσκηση 3 (Κεφάλαιο 11).

Η άσκηση ως λειτουργικότητα είναι πολύ απλή αλλά εισάγει τη δόμηση του προγράμματος με βάση ένα μενού επιλογής που δίνει τη δυνατότητα στον χρήστη να επιλέξει τη διεργασία που αυτός επιθυμεί να εκτελέσει η μηχανή. Αποτελεί καλή εξάσκηση για τις προτάσεις ελέγχου ροής και τη διαδικασία ανάπτυξης με την incremental development τεχνική. Αγνοήστε την Άσκηση 3β (παράγραφος 11.3.3)

switch: Πολλαπλή επιλογή

- Όταν βάσει μιας έκφρασης θέλουμε να επιλέξουμε ανάμεσα σε πολλές επιλογές, είναι πιο «βολική» η χρήση της **switch**.

```
switch (<έκφραση>)
{
  case <σταθ-εκφρ-1>: <προτ-1>; break;
  case <σταθ-εκφρ-2>: <προτ-2>; break;
      …
  case <σταθ-εκφρ-N>: <προτ-N>; break;
  default: <πρόταση>; break;
}
```

switch: Πολλαπλή επιλογή

- Κάθε <σταθ-έκφρ-i> πρέπει να είναι **μία τιμή int ή char** ή μία έκφραση μόνο με τέτοιες τιμές
- Δύο <σταθ-εκφρ-i> δεν μπορούν να έχουν την ίδια τιμή
- Αν <έκφραση>=<σταθ-εκφρ-x> τότε εκτελούνται όλες οι παρακάτω της x προτάσεις
  - ✓ Για να το αποτρέψουμε αυτό, χρειάζεται η **break**!
- Η <πρόταση> εκτελείται μόνο όταν καμμιά από τις <προτ-i> δεν ικανοποιείται
- Δεν υποστηρίζονται περιοχές τιμών (ranges), μόνο ισότητα
- Η default δεν είναι απαραίτητο να είναι στο τέλος

Παράδειγμα

```
switch (choice)   {
      case 1:
        x=a+b;
        break;
      case 2:
        x=a-b;
        break;
      case 3:
        x=a*b;
        break;
      case 4:
        x=a/b;
        break;
      default:
        printf("Ανύπαρκτη επιλογή");
        break;
}
```

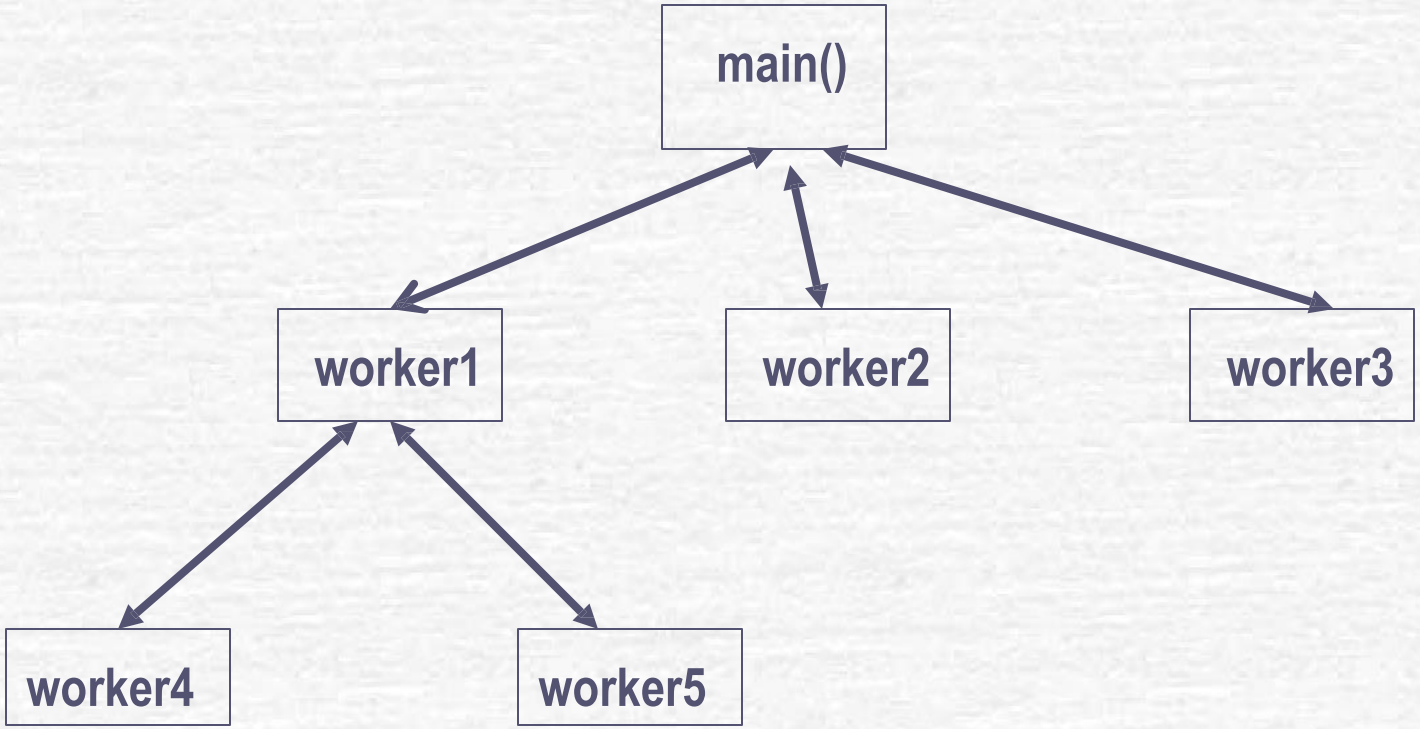
Παράδειγμα

```
switch (choice) {
      case 1: x=a+b; break;
      case 2: x=a-b; break;
      case 3: x=a*b;break;
      case  4:  x=a/b;break;
      default:
        printf("Ανύπαρκτη επιλογή");
        break;
}
```

```
/* Εκτέλεση λειτουργίας */
switch(choice){
        case 1 :
                /* Εισαγωγή των δύο αριθμών *
                        insert(a,b) …
                        break;
        case 'A' :
                /* πρόσθεση δύο αριθμών */
                        add(a,b) ...
                        break;
        …

        default :
                printf("Unknown command\n");
                break;
                }
```

Συναρτήσεις (Functions)



Υποπρογράμματα

- Αποτελούν τον τρόπο εφαρμογής (υλοποίησης) της τμηματοποίησης σε ένα πρόγραμμα.
- Κάθε υποπρόγραμμα είναι ένα αυτόνομο τμήμα (μικρό πρόγραμμα)
- Ο συνδυασμός των υποπρογραμμάτων συνιστά το (αρχικό) πρόγραμμα.
- Πλεονεκτήματα
  - Αποφυγή επαναλήψεων
  - Αύξηση επαναχρησιμοποίησης
  - Βελτίωση αναγνωσιμότητας
  - Ευκολότερη συντήρηση

Συναρτήσεις στη C

- Πρόγραμμα C = σύνολο συναρτήσεων
- Η συνάρτηση main αντιπροσωπεύει το κυρίως πρόγραμμα, δηλ. τον τρόπο με τον οποίο συνδυάζονται οι υπόλοιπες συναρτήσεις για τη λύση του προβλήματος.
- Μέρη συνάρτησης
  - Κεφαλίδα = η διεπαφή της συνάρτησης (όνομα, είσοδος, έξοδος)
  - Σώμα = υλοποίηση/ορισμός της συνάρτησης
- Προτάσεις συνάρτησης
  - Δήλωση – Κλήση - Ορισμός

Δήλωση Συνάρτησης

- Προσδιορίζεται η διεπαφή, ο τρόπος αναφοράς στη συνάρτηση:

  <τύπος> <όνομα – συν> ([<παράμετροι>]); Όπου

  <παράμετροι>:=<τύπος1> [<όνομ-παρ1>], …, <τύποςN> [<όνομ-παρN]

  τυπικά ορίσματα (ή είσοδοι)

  τυπικά αποτελέσματος (εξόδου)

- Αν η συνάρτηση δεν επιστρέφει κάποια τιμή, τότε χρησιμοποιείται σαν τύπος αποτελέσματος η λέξη κλειδί void.

Παραδείγματα

```
int max (int a, int b);

int min (int, int);

double exp ( double m, int n);

void swap (int a, int b);

char *getptr (char * str, char ch);

char *getptr (char str [ ], char ch);
```

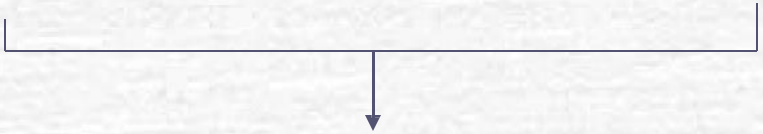
Παράδειγμα

```
#include <stdio.h>
int square(int y); /* αρχέτυπο συνάρτησης */
int main ( )
{
    int x;
    for (x=1; x<=10; x++)
            printf("%d", square(x));
    printf("\n");
    return 0;
}
/* ορισμός συνάρτησης */
int square( int y)
{
    return y * y;
}
```

Κλήση Συνάρτησης

- Καλείται η συνάρτηση για εκτέλεση με συγκεκριμένα ορίσματα

  <όνομα – συν> (<ορισ1>, <ορισ2>, …, <ορισN>);



πραγματικά ορίσματα (σταθερές, μεταβλητές, εκφράσεις|)

- Τα πραγματικά ορίσματα πρέπει να είναι του ίδιου αριθμού και τύπου με τα τυπικά ορίσματα

Παραδείγματα

```
swap ( x, y);

draw_circle (a/2.0, 2.0 * b, c);

max_num = max(num1, num2);

min_num = (num, 5);

x = y + max (num1/2.0, 3.0*num2);

printf ( “ο μέγιστος είναι: %d\n”, max(x1,x2));
```

Ορισμός Συνάρτησης

- Για κάθε μη ενσωματωμένη (δική μας) συνάρτηση πρέπει να ορίσουμε το σώμα της στο πρόγραμμα (μετά την main)

```
<τύπος> (<όνομα - συν> (<παράμετροι>)
        {
                <δηλώσεις τοπικών μεταβλητών>
                <προτάσεις>
        }
```

Παράδειγμα (1)

```
double embadon (double platos, double mikos)
{
double apotelesma;
apotelesma = platos * mikos;
platos=15;
 return (apotelesma);
}
```

τυπικές παράμετροι

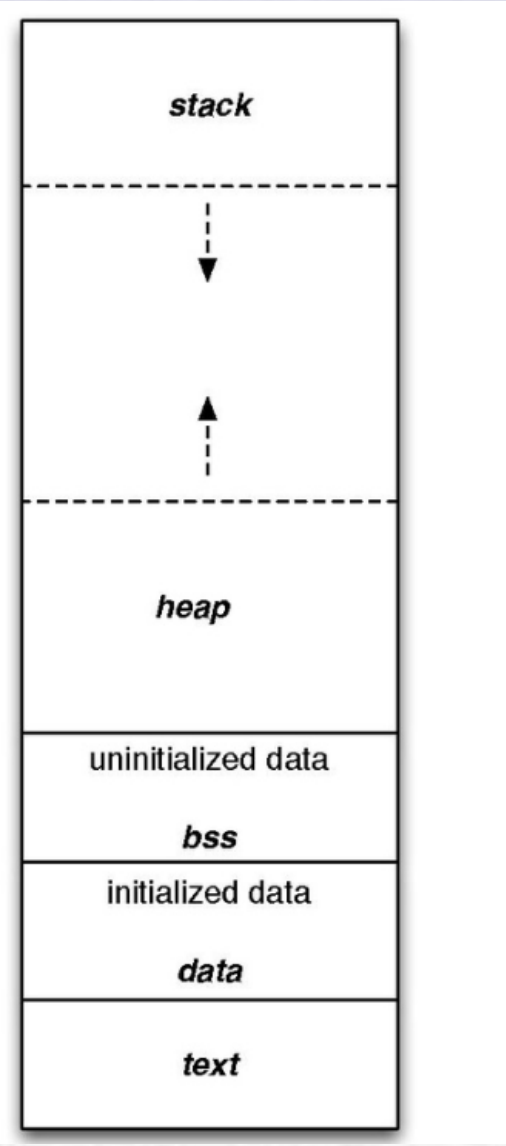
τοπική μεταβλητή

Επιστροφή ελέγχου και τιμής στην καλούσα συνάρτηση

Παράδειγμα (2)

```
int max (int a, int b )
{
    int max;
    max = ( a > b) ? a : b;
    return (max);
}

int max( int a, int b)
{
    return ((a > b) ? a : b);
}
```



Memory Layout of a C program

https://commons.wikimedia.org/wiki/File:Program_memory_layout.pdf

Εμβέλεια Μεταβλητών

- Εμβέλεια = το τμήμα του προγράμματος στο οποίο έχει ισχύ ή είναι ορατή η μεταβλητή.
- Διάρκεια ζωής= πόσο υπάρχει στη μνήμη.

- Πότε μας αφορά η εμβέλεια;

  Όταν έχουμε μεταβλητές με το ίδιο όνομα.

Τύποι Εμβέλειας (1)

- Εξωτερικές μεταβλητές

  - Γενικές ή καθολικές μεταβλητές (global variables)
    - Δηλώνονται έξω και πάνω από κάθε συνάρτηση (και την main), στο τμήμα δηλώσεων μεταβλητών.
    - Η εμβέλειά τους είναι όλο το αρχείο του πηγαίου κώδικα που

    ανήκει η δήλωση, από το σημείο της δήλωσης και κάτω (εμβέλεια αρχείου

Τύποι Εμβέλειας (2)

- Εσωτερικές ή αυτόματες μεταβλητές
  - Τοπικές μεταβλητές (local variables)
    - Δηλώνονται μέσα σε μία συνάρτηση.
    - Η εμβέλειά τους είναι το σώμα της συνάρτησης (εμβέλεια μπλοκ)
  - Τυπικές παράμετροι συναρτήσεων
    - Η εμβέλειά τους είναι το σώμα της συνάρτησης (εμβέλεια μπλοκ)
  - Μεταβλητές σύνθετης πρότασης
    - Δηλώνονται μέσα σε μία σύνθετη πρόταση
    - Η εμβέλειά τους είναι το σημείο της δήλωσης μέχρι το τέλος της σύνθετης πρότασης (εμβέλεια μπλοκ).
  - Στατικές Μεταβλητές

- Δηλώνονται σε κάποια (ες) συνάρτηση (εις) (μετά την main) με τη λέξη κλειδί static πριν από τον τύπο μεταβλητής.

-Η εμβέλειά τους είναι το αρχείο του πηγαίου κώδικα που ανήκει η δήλωση, από το σημείο της δήλωσης και κάτω (εμβέλεια αρχείου)

Κανόνες Εμβέλειας Μεταβλητών

- Μεταβλητές με το ίδιο όνομα επιτρέπονται μόνο όταν έχουν διαφορετική εμβέλεια
- Μεταβλητή με μικρότερη εμβέλεια αποκρύπτει πιθανώς ομώνυμες μεταβλητές μεγαλύτερης εμβέλειας.

Εμβέλεια Συνάρτησης

- Οι συναρτήσεις, όπως και οι μεταβλητές έχουν εμβέλεια
- Η εμβέλεια μίας συνάρτησης εκτείνεται από το σημείο της δήλωσής της μέχρι το τέλος του προγράμματος.
- Αν μία συνάρτηση δηλωθεί static, τότε η εμβέλειά της περιορίζεται το αρχείο που δηλώθηκε.

- Μεταβλητή με μικρότερη εμβέλεια αποκρύπτει πιθανώς ομώνυμες μεταβλητές μεγαλύτερης εμβέλειας.

Διάρκεια Μεταβλητής

- Ο χρόνος δέσμευσης της μνήμης που περιέχει την τιμή της μεταβλητής
- Καθολική μεταβλητή: διάρκεια εκτέλεσης προγράμματος (πλήρης διάρκεια).
- Τοπική μεταβλητή: διάρκεια εκτέλεσης συνάρτησης (περιορισμένη διάρκεια)
- Τυπική παράμετρος: διάρκεια εκτέλεσης συνάρτησης (περιορισμένη διάρκεια)
- Στατική τοπική μεταβλητή: διάρκεια εκτέλεσης προγράμματος (πλήρης διάρκεια)

Αρχικοποίηση Μεταβλητών

- Μία τοπική μεταβλητή περιορισμένης διάρκειας αρχικοποιείται με κάθε είσοδο στο μπλοκ (συνάρτηση) που ορίζεται
- Μία τοπική μεταβλητή πλήρους διάρκειας αρχικοποιείται με την έναρξη εκτέλεσης του προγράμματος
- Π.χ. static int num;

```
func (int) {
static int count = 0;
int num =100; ... }
```

Πέρασμα Παραμέτρων

- Κατ΄αξία ή τιμή (by value)

  Η συνάρτηση δουλεύει σε αντίγραφα των πραγματικών παραμέτρων

- Κατ’ αναφορά (by reference)

  Η συνάρτηση δουλεύει στις πραγματικές παραμέτρους (μόνο για πίνακες)

Πολλαπλά αρχεία

-Σε πολλαπλά αρχεία υπάρχει ΜΟΝΟ ένα αρχείο με τη main()
-Μία εξωτερική μεταβλητή μπορεί να χρησιμοποιηθεί από περισσότερα από ένα αρχεία αν και μόνο εάν η δήλωσή της στα άλλα αρχεία συνοδεύεται από keyword extern
-Μία συνάρτηση θα πρέπει να δηλωθεί σε κάθε αρχείο που τη χρησιμοποιεί, ορίζεται όμως ΜΟΝΟ σε ένα

-Αν μία συνάρτηση ή μεταβλητή δηλωθεί static, τότε η εμβέλειά της περιορίζεται αναγκαστικά στο αρχείο που δηλώθηκε.

Συνήθεις Πρακτικές

(λ.χ. http://stackoverflow.com/questions/1433204/how-do-i-use-extern-to-share-variables-between-source-files-in-c)

- Ένα header file περιέχει μόνο extern δηλώσεις μεταβλητών — ποτέ static
- Για κάθε μεταβλητή μόνο ένα header file τη δηλώνει (SPOT — Single Point of Truth). Αυτό το αρχείο κεφαλίδα συμπεριλαμβάνεται από το ΜΟΝΑΔΙΚΟ ΑΡΧΕΙΟ ΠΟΥ ΟΡΙΖΕΙ ΤΗ ΜΕΤΑΒΛΗΤΗ ΚΑΙ ΑΠΟ ΚΑΘΕ ΑΡΧΕΙΟ ΠΟΥ ΤΗΝ ΑΝΑΦΕΡΕΙ. Συνεπώς για κάθε μεταβλητή ΕΝΑ ΚΑΙ ΜΟΝΟ ΕΝΑ ΑΡΧΕΙΟ ΚΕΦΑΛΙΔΑ ΤΗ ΔΗΛΩΝΕΙ ΚΑΙ ΕΝΑ ΚΑΙ ΜΟΝΟ ΕΝΑ ΑΡΧΕΙΟ ΤΗΝ ΟΡΙΖΕΙ.
- Ένα αρχείο πηγαίου κώδικα δεν περιέχει extern δηλώσεις μεταβλητών. — τα αρχεία πηγαίου κώδικα πάντα περιλαμβάνουν το (μόνο) αρχείο κεφαλίδα που εμπεριέχει τις extern δηλώσεις. Το αρχείο που ορίζει τη μεταβλητή την αρχικοποιεί επίσης. Η αρχικοποίηση με 0 είναι απαραίτητη.
- Αποφύγετε ολικές μεταβλητές όπου είναι εφικτό, χρησιμοποιήστε συναρτήσεις.

Η Βιβλιοθήκη i2p

1. Βιβλιοθήκη τρίτου κατασκευαστή (Third-Party library)

Third-Party library είναι μια βιβλιοθήκη που έχει αναπτυχθεί από άλλον προγραμματιστή. Η σημασία αξιοποίησης βιβλιοθήκης τρίτου κατασκευαστή είναι εμφανής από το παρακάτω απόσπασμα.

"In computer programming, a **third-party software component** is a reusable software component developed to be either freely distributed or sold by an entity other than the original vendor of the development platform. The third-party software component market thrives because many programmers believe that **component-oriented development** improves the efficiency and the quality of developing custom applications." Wikipedia

2. Αντικείμενο και Στόχος της βιβλιοθήκη i2p

Η βιβλιοθήκη i2p αναπτύχθηκε για δύο λόγους:

1. για να σας δώσει τη δυνατότητα να εξοικειωθείτε με τη διαδικασία αξιοποίησης βιβλιοθήκης τρίτου κατασκευαστή, και,
2. να σας διευκολύνει στην ανάπτυξη των πρώτων εκδόσεων ορισμένων προγραμμάτων.

3. Αξιοποίηση βιβλιοθήκης

Για να αξιοποιήσετε μια βιβλιοθήκη θα πρέπει να κάνετε τις παρακάτω ενέργειες:

Ε1-Κατεβάστε την βιβλιοθήκη

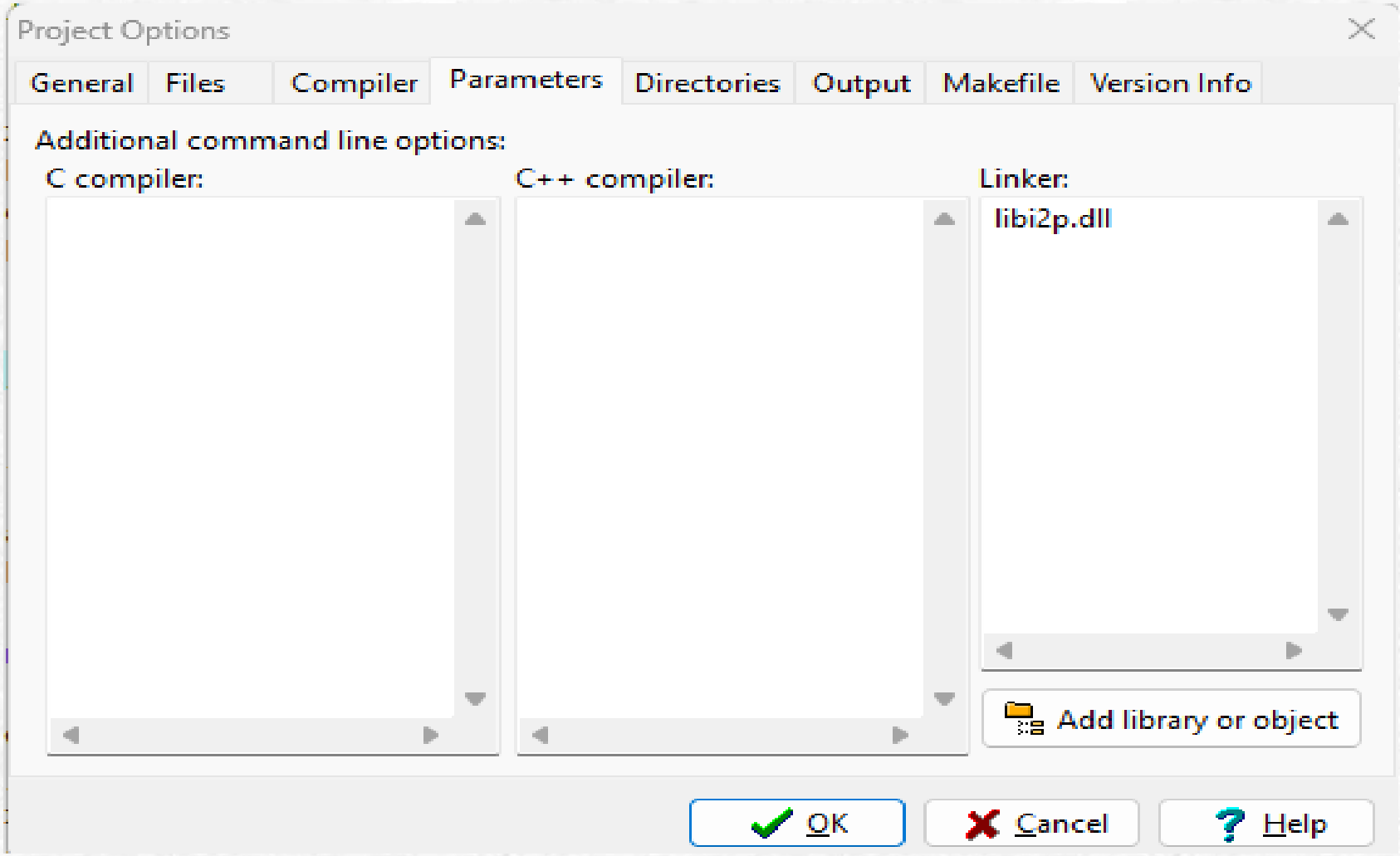
Κατεβάστε και τοποθετήστε την βιβλιοθήκη (αρχείο .h και αρχείο .dll) στο ευρετήριο του project σας.

Ε2-Συμπεριλάβετε το αρχείο επικεφαλίδας (#include)

Συμπεριλάβετε με την εντολή προεπεξεργαστή #include στο αρχείο πηγαίου κώδικα σας το αρχείο επικεφαλίδας, π.χ. #include “i2p.h”.
Τοποθετούμε το header file σε “ ” για να ψάξει ο προεπεξεργαστής (pre-processor) το αρχείο στο φάκελο (ευρετήριο) του προγράμματος μας.
Σε < > περικλείονται τα αρχεία επικεφαλίδας της βασικής βιβλιοθήκης για να δηλώσουν στον επεξεργαστή πως αυτά θα τα αναζητήσει στον φάκελο του compiler που χρησιμοποιείτε.

Ε3-Ενημερώστε το IDE σας για το αρχείο .dll

Για παράδειγμα για να ενημερώσετε το DevC++ επιλέξτε
Project->Projects Options>Parameters και
στην συνέχεια `Add library or Object` (Σχήμα 1). Επιλέξτε από το ευρετήριο του project το αρχείο της βιβλιοθήκης. Θα το δείτε στο πλαίσιο κάτω από τον Linker καθώς το αρχείο αυτό θα το αξιοποιήσει, μετά την μεταγλώττιση, ο Linker. Επιλέξτε OK.



Ε4-Καλέστε συνάρτηση της βιβλιοθήκης

Καλέστε μια συνάρτηση της βιβλιοθήκης πάντα σύμφωνα με το function prototype της που θα βρείτε στο αρχείο επικεφαλίδας της βιβλιοθήκης. Στην επόμενη ενότητα δίνονται οι συναρτήσεις που περιέχει η βιβλιοθήκη και στην μεθεπόμενη οδηγίες και παραδείγματα κλήσης των συναρτήσεων.

Δισδιάστατος Πίνακας (1)

- Αναπαράσταση 10 φοιτητών και 4 μαθημάτων: χρήση δισδιάστατου πίνακα.
  - int grades[10][4]
  - πίνακας 10 στοιχείων, κάθε στοιχείο πίνακας 4 ακεραίων
  - ή πίνακας 10 γραμμών και 4 στηλών.

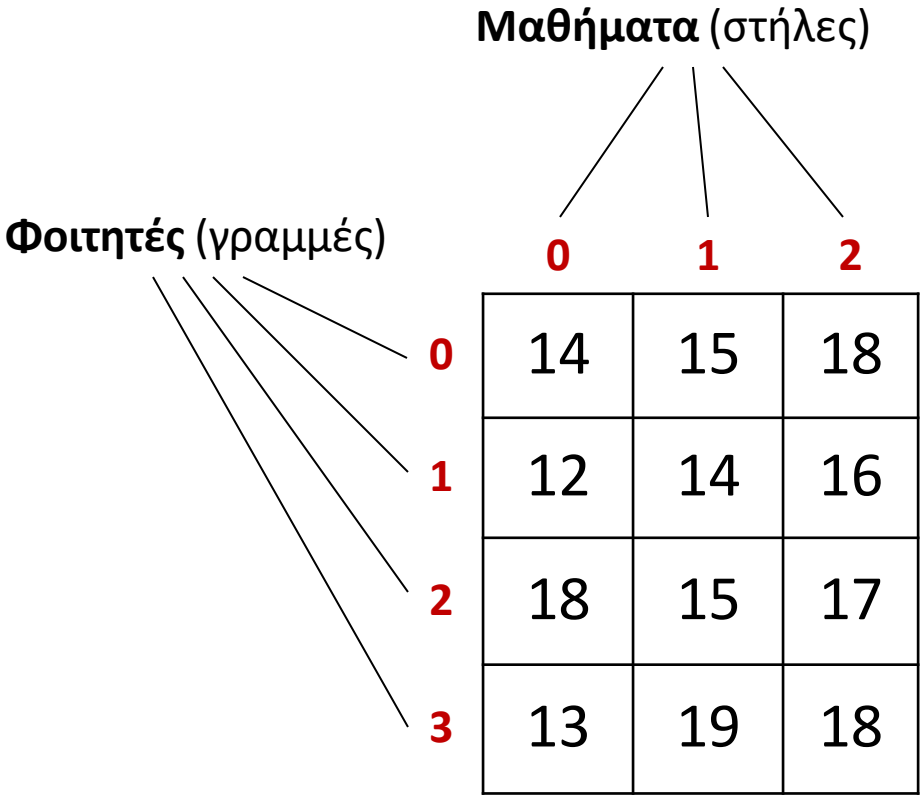
- grades[0][1] -> βαθμός 2ου μαθήματος, πρώτου φοιτητή

- Αποθήκευση στη μνήμη κατά γραμμές (σαν μονοδιάστατος)

Πέρασμα πίνακα δύο διαστάσεων σε συνάρτηση)

```
#include <stdio.h>
#include <stdlib.h>
double Average(int table[][5], int rows); /*Πίνακας 2 διαστάσεων */
/* Η δεύτερη (και οι επόμενες) διάσταση του πίνακα δίνεται ρητά */
main(){
   /* Αρχικοποίηση πίνακα */
   int table[2][5] = {{3, 1, 15, 0, 16},{12, 5, 16, 6, 1}};
   double avg;
/* Μεταβιβάζουμε τον πίνακα στη συνάρτηση ως δείκτη, καθώς και τη 2η
 διάσταση */
   avg = Average(table, 2);
   printf( "The average is: %f ", avg );
}
double Average(int table[][5], int rows){
  int i, j;
  double sum=0.0;
  for (i=0; i<rows; i++)
      for (j=0; j<5; j++)
            sum += table[i][j];
  return sum/(rows*5);
}
```

Δισδιάστατος Πίνακας (2)




```
// Αρχικοποίηση
int grades[4][3]=
{
  {14,15,18},
  {12,14,16},
  {18,15,17},
  {13,19,18}
}
```

Πίνακας λέξεων

- Για να αποθηκεύσουμε αλφαριθμητικά χρησιμοποιούμε πίνακα δύο διαστάσεων χαρακτήρων.

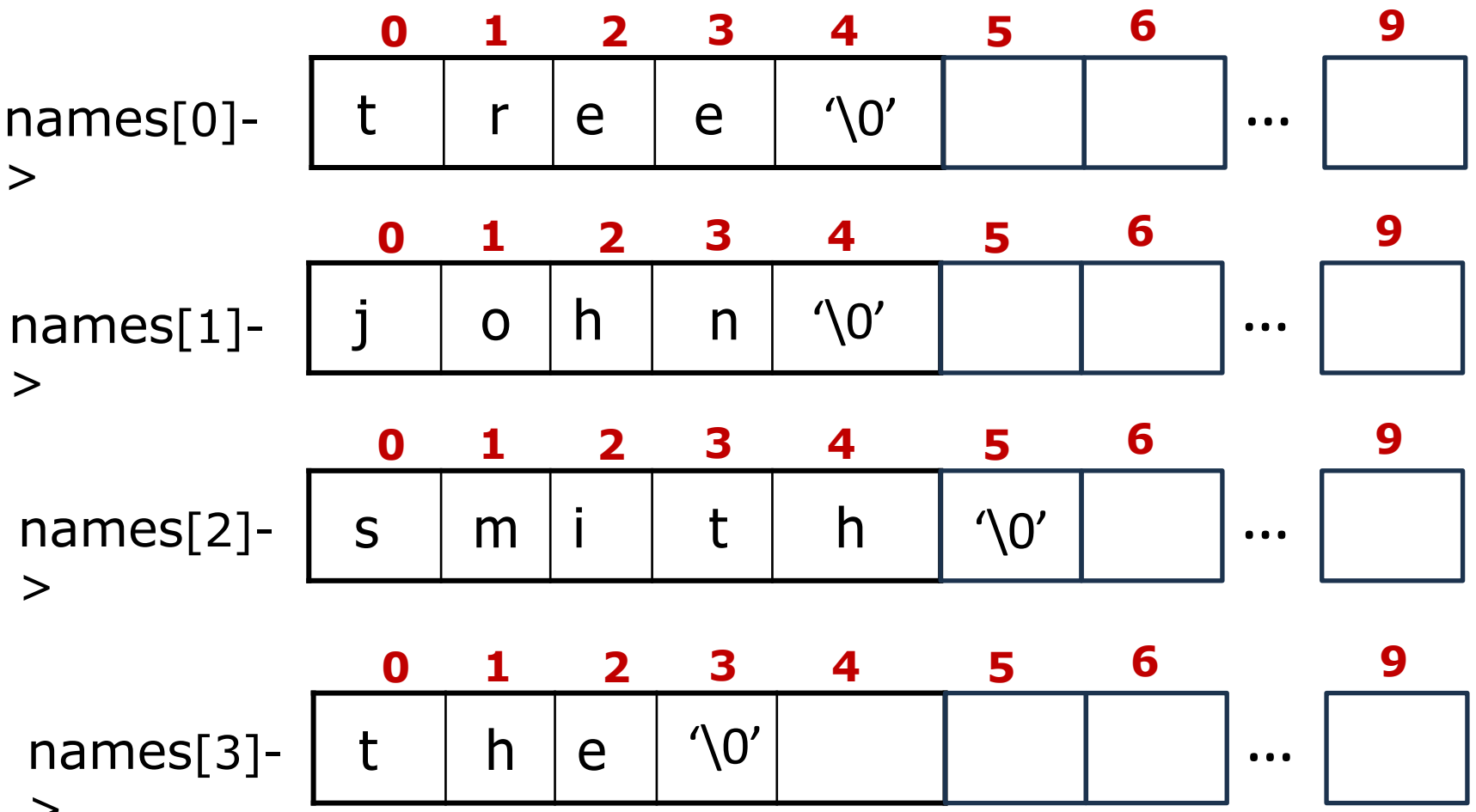
  Για παράδειγμα:

```
char name[4][10];
char names[4][10]={“tree“,”john”,“smith”, “the”}
```

`names[i][j]` -> ο χαρακτήρας που αποθηκεύεται στην i-γραμμή και j-στήλη

`names[i]` ->

καθένα από τα Ν στοιχεία names[0],..., names[3] μπορούν να προσπελαστούν σαν μονοδιάστατος πίνακας (διεύθυνση κελιού πίνακα που περιέχει τα Μ στοιχεία της αντίστοιχης γραμμής).



Εναλλακτικά:

```
char * names={“tree“,”john”,“smith”, “the”}
```

Ιδέες για εξάσκηση

- Πολλαπλασιασμός πινάκων
- Υλοποίηση λεξικού μετάφρασης όρων

Γράψτε ένα πρόγραμμα το οποίο θα δέχεται ως είσοδο ένα πίνακα αλφαριθμητικών. Κάθε αλφαριθμητικό περιέχει ένα σύνολο από λέξεις χωρισμένα από ένα ή περισσότερα κενά (μπορείτε για απλότητα να θεωρήσετε ότι κάθε πρόταση είναι μία λέξη, έχει όμως νόημα για εξάσκηση να κάνετε κώδικα και για την περίπτωση που κάθε πρόταση έχει πιο πολλές από μία λέξεις). Το πρόγραμμα θα μετράει το μήκος των λέξεων αυτών και θα εμφανίζει το ιστόγραμμα των μηκών με γραφικό τρόπο ως εξής:

```
Word Length                         |Number of Occurrences

1                                   |

2                                   | *******

3                                   | **********

4                                   | ****

5                                   | ***

6                                   | *****

7                                   | ***

8                                   | ***

9                                   | *

10                                  |

11                                  |

12                                  |

13                                  |

diogenis>
```