ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

2η Εβδομάδα: Στοιχεία της Γλώσσας C

Αναφορές σχετικά με το Υλικό Παραδόσεων

Οι διαφάνειες της διάλεξης στηρίζονται, εν μέρει, σε υλικό παραδόσεων παλαιότερων ετών του **Τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογία Υπολογιστών του Πανεπιστημίου Πατρών** καθώς και του **Τμήματος Πληροφορικής του Πανεπιστήμιο Κύπρου**

Ανασκόπηση της 1ης Εβδομάδας

- Βασικά στοιχεία της Γλώσσας Προγραμματισμού C
- Τύποι Δεδομένων
- Δομές επιλογής και δομές επανάληψης
- Συναρτήσεις και χρήση των συναρτήσεων
- Παραδείγματα προετοιμασίας για την πρώτη εργαστηριακή άσκηση

Προγραμματισμός

Περιγραφή του Προβλήματος: *Διαδικασία κατανόησης του τι θέλουμε να κάνει το πρόγραμμα*

Προγραμματισμός: Η διαδικασία του να **επινοήσουμε αλγόριθμο** και να **συντάξουμε πρόγραμμα**.

Πρόγραμμα: Αλγόριθμος μεταφρασμένος σε μορφή προγραμματιστικών εντολών με τη χρήση της **γλώσσας προγραμματισμού** C

Μεταγλώττιση (compilation)

Εκτελέσιμο αρχείο: Πρόγραμμα μεταφρασμένο σε μορφή κατανοητή από τον Η/Υ.

Υπολογιστικό Πρόβλημα

Δίνεται ακέραιος x. ***Να υπολογιστεί*** το τετράγωνό του, x^2.

Αλγόριθμος

1. Διάβασε x.
2. Υπολόγισε x^2
3. Τύπωσε x^2.

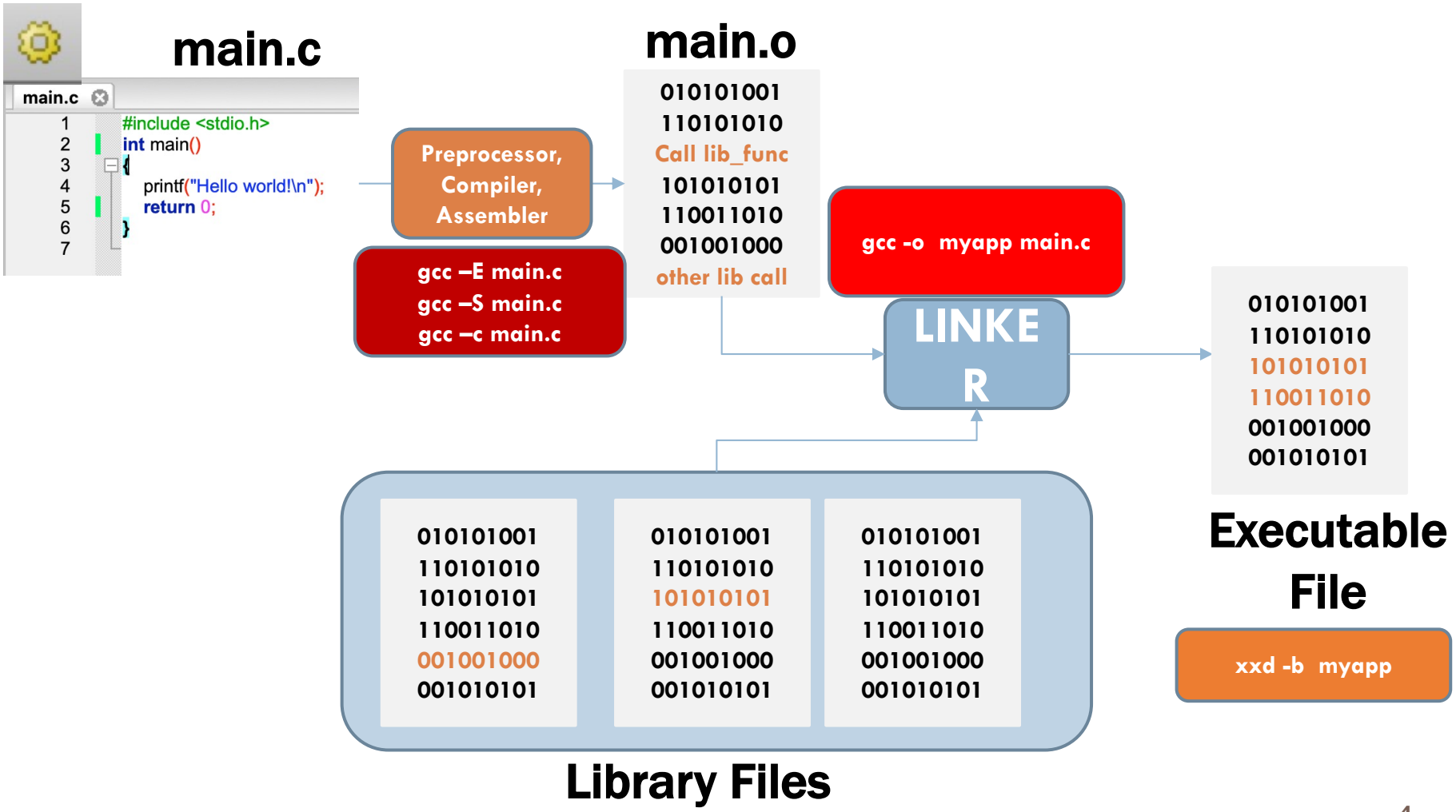
Πρόγραμμα

```
int x, square;
printf("Give me x:");
scanf("%d", &x);
square = x * x;
printf("Square of x is %d!", square);
```

Εκτελέσιμο Αρχείο

```
00101010010101111011101
01010110100100100110101
01010001010100
```

Μεταγλώτισση στη C



Έλεγχος και διόρθωση λαθών

3 κατηγορίες λαθών που μπορεί να δημιουργηθούν:

- **Συντακτικά λάθη (syntax errors) → Αν υπάρχουν συντακτικά λάθη στον κώδικα δεν παράγεται το εκτελέσιμο αρχείο.**
- **Λογικά λάθη (bugs) → Εντοπίζονται κατά την εκτέλεση του προγράμματος**
- **Run-time Errors (Σφάλματα που παρουσιάζονται μόνο κατά την εκτέλεση του προγράμματος)**

```
void main(void) {
            float side_a, 8side_b, hypotenuse;

            printf("Side a:")
            scanf("%f", &side_a);
            printf("Side b:");
            scanf("%f", &8side_b);

            hypotenuse = sqrt(side_a * side_a + 8side_b * 8side_b);

            printf("The hypotenuse is %f", hypotenuse);
}
```

```
void main(void) {
            float side_a, side_b, hypotenuse;

            printf("Side a:")
            scanf("%f", &side_a);
            printf("Side b:");
            scanf("%f", &side_b);

            hypotenuse = sqrt(side_a * side_a * side_b * side_b);

            printf("The hypotenuse is %f", hypotenuse);
}
```

```
void main(void) {
            float side_a, side_b, hypotenuse;

            printf("Side a:")
            scanf("%f", &side_a);
            printf("Side b:");
            scanf("%f", &side_b);

            hypotenuse = sqrt(side_a * side_a * side_b * side_b);

            printf("The hypotenuse is %f", hypotenuse/0);
}
```

Δήλωση Σταθεράς Σε επίπεδο Προεπεξεργαστή και Μεταγλωττιστή

- Με την εντολή **#define** του προεπεξεργαστή ορίζουμε συμβολοσειρές οι οποίες αντικαθίστανται με τις αντίστοιχες τιμές κατά την εκτέλεση

- Το **const** λέει στον μεταγλωττιστή ότι η συγκεκριμένη μεταβλητή δεν θα αλλάξει ποτέ τιμή

- Αν προσπαθήσουμε να αλλάξουμε τιμή σε μεταβλητή δηλωμένη ως const, θα προκύψει λάθος μεταγλώττισης

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

#define PI 3.14
int main()
{
    const float Plc=3.14;
    printf("PI %.2f  \n", PI);
    printf("PI %.2f  \n", Plc);
    return 0;
}
```

Παράδειγμα

```c
#include <stdio.h>
int main() {
  int i, num, sum=0;

  for (i=0; i<10; i++) {
    scanf("%d", &num);
    sum = sum + num;
    printf("partial sum: %d\n", sum);
  }

  printf("total: %d", sum);

  return 0;
}
```

```c
#include <stdio.h>
#define N 10
int main() {
  int i, num, sum=0;

  for (i=0; i<N; i++) {
   scanf("%d", &num);
   sum = sum + num;
   printf("partial sum: %d\n", sum);
  }

  printf("total: %d", sum);

  return 0;
}
```

Μετατροπή τύπων - Casting

int x = 5;

float y = x / 3;

Ο τύπος της έκφρασης x / 3 είναι int

Η τιμή της έκφρασης είναι ακέραια τιμή 1

Θα μετατραπεί σε float και θα ανατεθεί στο y η τιμή 1.0

float y = (float) (x / 3);

Η τιμή του y θα είναι 1.0

Εάν το ζητούμενο είναι να είναι ο τύπος της εκφράσης float τότε προχωρούμε σε **ρητή μετατροπή τύπων**:

float y = (float) x / 3; ή

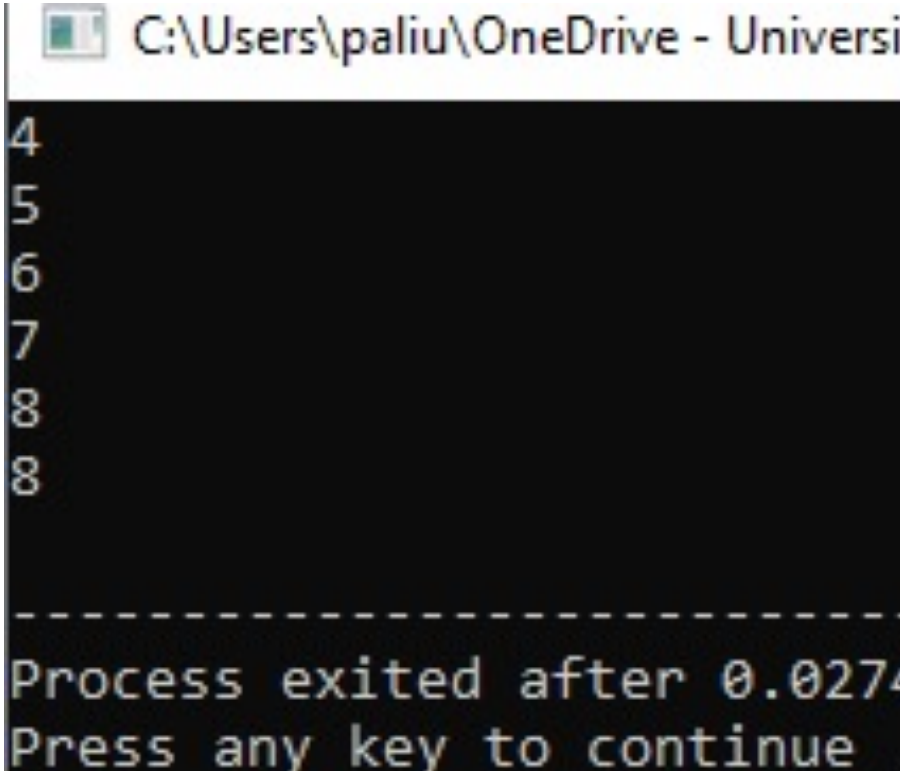
float y = x / (float) 3; ή

y = x / 3.0 ;

Η τιμή του y θα είναι 1.666667

Τι τυπώνεται;

```c
#include <stdio.h>

int main() {

    int a = 4;

    printf("%d\n", a);
    a++;
    printf("%d\n", a);
    ++a;
    printf("%d\n", a++);
    printf("%d\n", a);
    printf("%d\n", ++a);
    printf("%d\n", a);

    return 0;
}
```



Παράδειγμα

Γράψετε ένα πρόγραμμα το οποίο να ρωτάει το χρήστη για το βαθμό του και να του τυπώνει στην οθόνη:

- **A – Excellent**
- **B και C - Well done**
- **D - You passed**
- **F - Better try again**

Διαφορετικά να τυπώνει Invalid grade

```c
#include <stdio.h>

int main () {

   char grade;

   printf("Give me your Grade (A/B/C/D/F): ");
   scanf("%c", &grade);

   switch(grade) {
      case 'A' :
         printf("Excellent!\n" );
         break;
      case 'B' :
      case 'C' :
         printf("Well done\n" );
         break;
      case 'D' :
         printf("You passed\n" );
         break;
      case 'F' :
         printf("Better try again\n" );
         break;
      default :
         printf("Invalid grade\n" );
   }

   printf("Your grade is  %c\n", grade );
}
```

Σύνδεση εκφράσεων με λογικούς τελεστές

```c
#include <stdio.h>

int main() {
  int a = 1, b = 0;

  if (a==1 || ++b ==1) {
          printf("hello\n");
  }
  printf("value of b after if: %d\n", b);

     return 0;
}
```

Τι αλλάζει στη συμπεριφορά,

αν το a αρχικοποιηθεί στην τιμή

0;

Short-circuit evaluation

Στόχοι

- Ο σκοπός της 2ης εβδομάδας είναι η περαιτέρω εξοικείωση σας με τα στοιχείς της Γλώσσας C

- Θα αναφερθούμε σε **παραδείγματα** ώστε να εμπεδώσουμε καλύτερα τις θεωρητικές γνώσεις

- Θα αναφερθούμε στη διαχείριση της μνήμης από τη σκοπιά της εκτέλεσης ενός προγράμματος

Εμβέλεια Μεταβλητών

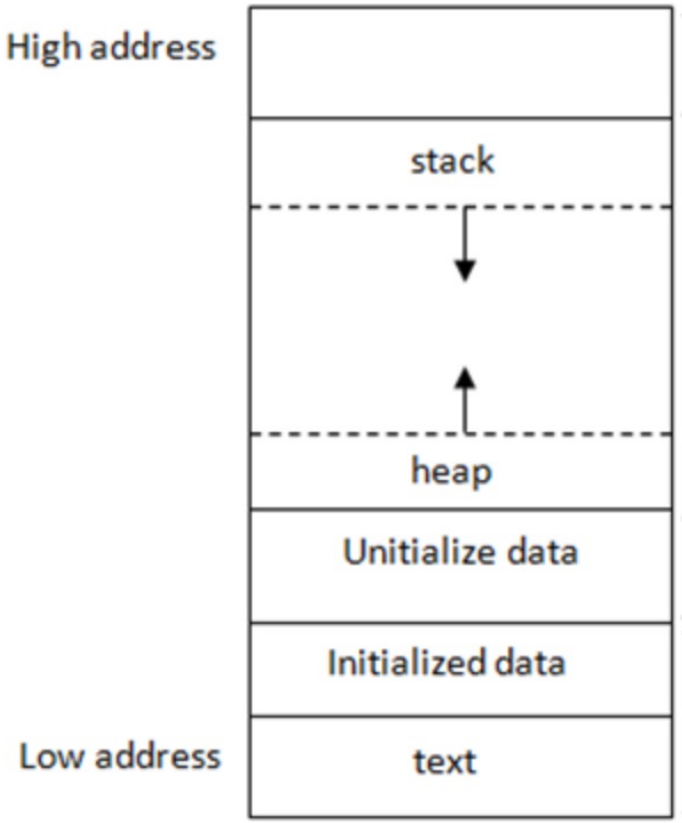
- Οι παράμετροι μιας συνάρτησης θεωρούνται τοπικές της μεταβλητές και συνεπώς είναι γνωστές μόνο στο μπλοκ της συνάρτησης

```
#include <stdio.h>
#include <stdlib.h>
int max (int, int);
int min (int,int);
int main(void)
{
    int a,b;

    printf("\n Please enter two integers : ");
    scanf("%d %d", &a,&b);
    if(a!=b) printf("\n max is %d and min is %d \n", max(a,b), min (a,b));
    return 0;
}

int max(int x, int y){
 return (x>y?x:y);
}

int min(int x, int y){
 return (x<y?x:y);
}
```

Διάταξη Μνήμης

Κάθε πρόγραμμα που εκτελείται έχει τη δική του διάταξη μνήμης, χωρισμένη σε πολλά τμήματα, όπως:

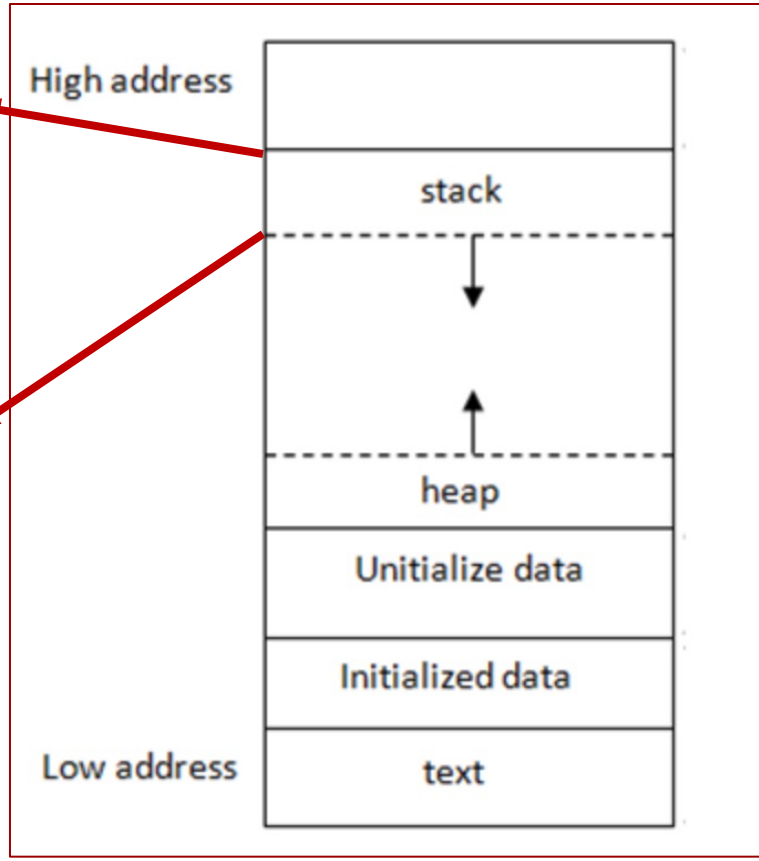
- **stack**: αποθηκεύει τοπικές μεταβλητές
- **heap**: δυναμική μνήμη για τη δέσμευση και αποδέσμευση μνήμης κατά τη διάρκεια εκτέλεσης
- **data**: αποθηκεύει global variables, σε initialized και uninitialized
- **text**: αποθηκεύει τις προς εκτέλεση εντολές



```c
#include <stdio.h>
#include <stdlib.h>

void foo(void);
void foo2(void);

int main()
{
    foo();
    foo2();

    return 0;
}

void foo(void){
    int x,y;
    char z;
    printf("Address of int x (foo1): %lld\n", &x);
    printf("Address of int y (foo1): %lld\n", &y);
    printf("Address of char z(foo1): %lld\n", &z);
}

void foo2(void){
    int x,y;
    char z;
    printf("\n Address of int x (foo2): %lld\n", &x);
    printf("Address of int y (foo2): %lld\n", &y);
    printf("Address of char z(foo2): %lld\n", &z);
}
```

```
Address of int x (foo1): 140732696594956
Address of int y (foo1): 140732696594952
Address of char z(foo1): 140732696594951

 Address of int x (foo2): 140732696594956
Address of int y (foo2): 140732696594952
Address of char z(foo2): 140732696594951
```

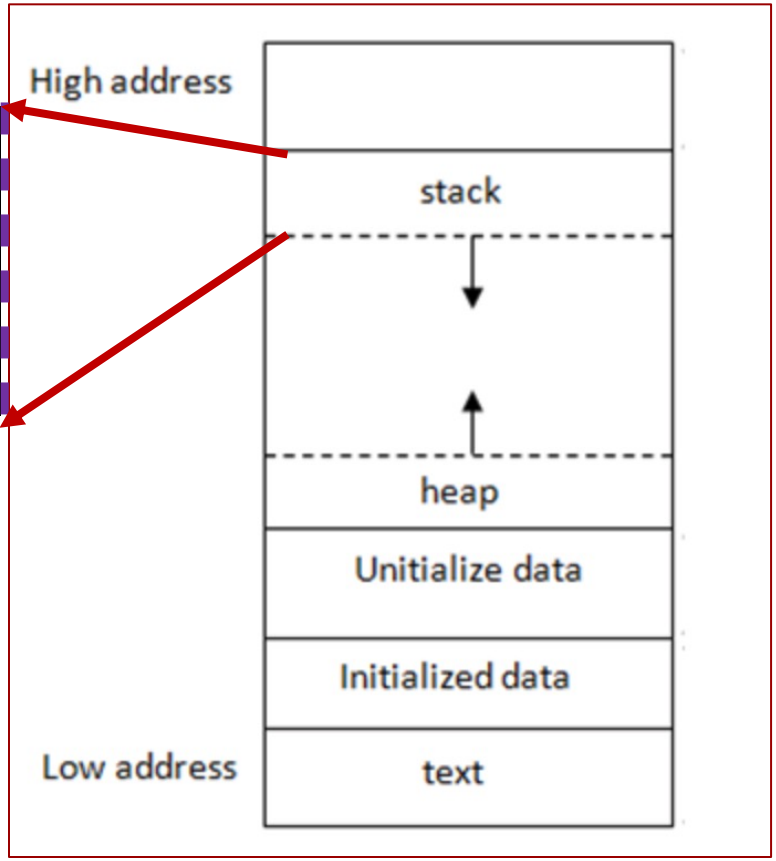
Τι παρατηρείτε;



```c
#include <stdio.h>
#include <stdlib.h>

void foo(void);
void foo2(void);

int main()
{
    foo();
    return 0;
}

void foo(void){
    int x,y;
    char z;
    printf("Address of int x (foo1): %lld\n", &x);
    printf("Address of int y (foo1): %lld\n", &y);
    printf("Address of char z(foo1): %lld\n", &z);
    foo2();
}

void foo2(void){
    int x,y;
    char z;
    printf("\nAddress of int x (foo2): %lld\n", &x);
    printf("Address of int y (foo2): %lld\n", &y);
    printf("Address of char z(foo2): %lld\n", &z);
}
```

```
Address of int x (foo1): 140732710824460
Address of int y (foo1): 140732710824456
Address of char z(foo1): 140732710824455

 Address of int x (foo2): 140732710824428
Address of int y (foo2): 140732710824424
Address of char z(foo2): 140732710824423
```

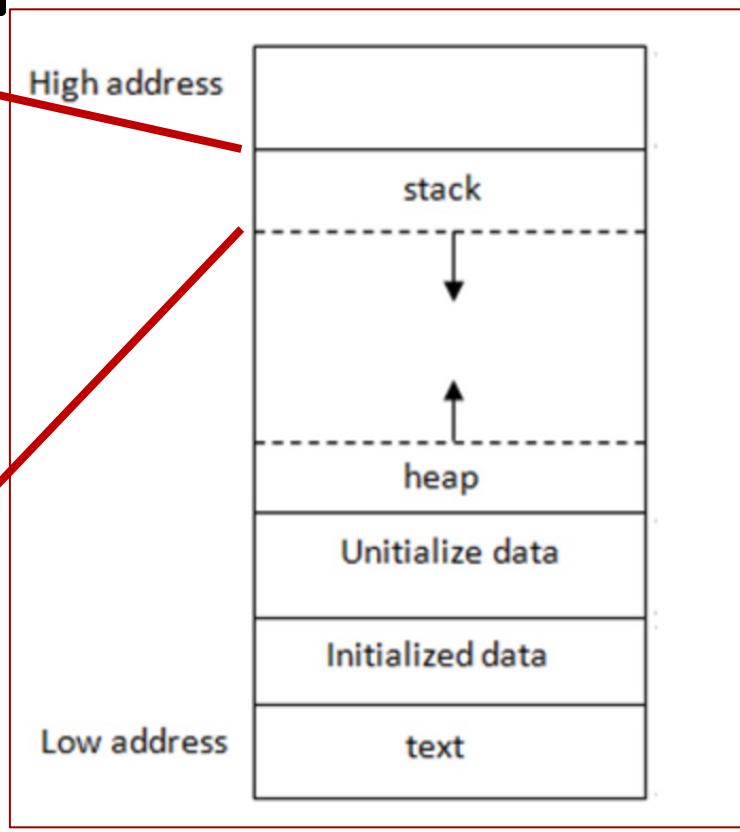
Τι παρατηρείτε;



Τι παρατηρείτε;

```c
#include <stdio.h>
#include <stdlib.h>

void foo(void);
void foo2(void);

int main()
{
    foo();
    foo2();
    return 0;
}

void foo(void){
    int x,y;
    char z;
    printf("Address of int x (foo1): %lld\n", &x);
    printf("Address of int y (foo1): %lld\n", &y);
    printf("Address of char z(foo1): %lld\n", &z);
    foo2();
}

void foo2(void){
    int x,y;
    char z;
    printf("\nAddress of int x (foo2): %lld\n", &x);
    printf("Address of int y (foo2): %lld\n", &y);
    printf("Address of char z(foo2): %lld\n", &z);
}
```

```
Address of int x (foo1): 140732700031500
Address of int y (foo1): 140732700031496
Address of char z(foo1): 140732700031495

Address of int x (foo2): 140732700031468
Address of int y (foo2): 140732700031464
Address of char z(foo2): 140732700031463

Address of int x (foo2): 140732700031500
Address of int y (foo2): 140732700031496
Address of char z(foo2): 140732700031495
```



Παράδειγμα

- Να γραφεί ένα πρόγραμμα που διαβάζει **ακεραίους** έναν κάθε φορά και τυπώνει το μερικό άθροισμα **των άρτιων**, **όσο** ο χρήστης δίνει ως είσοδο **αριθμούς > 0**.
- Αριθμοί ≤0 δεν λαμβάνονται υπόψη στους υπολογισμούς
- Στο τέλος τυπώνεται το συνολικό άθροισμα και το γινόμενό **των άρτιων**.

έκδοση 1

```c
#include <stdio.h>
int main( )
{

  int input, sum = 0 , prod = 1;

  scanf("%d", &input);

  while (input>0) {
        sum = sum + input ;
        printf("partial sum: %d\n", sum);
        prod = prod * input ;
        scanf("%d", &input);
    }

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

  return 0;
}
```

έκδοση 2

```c
#include <stdio.h>

int main( )
{

  int input, sum = 0 , prod = 1;

  for (scanf("%d", &input); input>0; scanf("%d", &input) )
{
        sum = sum + input ;
        printf("partial sum: %d\n", sum);
        prod = prod * input ;
      }

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

 return 0;
}
```

έκδοση 3

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{

  int input, sum = 0 , prod = 1;

  do {
     scanf("%d", &input);
     if (input >0 ) {
        sum = sum + input ;
        printf("partial sum: %d\n", sum);
        prod = prod * input ;
        }
     } while (input>0) ;

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

return 0;
}
```

έκδοση 4

```c
#include <stdio.h>

int main()
{

  int input, sum = 0 , prod = 1;

  do {
     scanf("%d", &input);
     if (input <=0 )
        break;
     sum = sum + input ;
     printf("partial sum: %d\n", sum);
     prod = prod * input ;
     } while (input>0) ;

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

return 0;
}
```

έκδοση 4α

```c
#include <stdio.h>

int main()
{

  int input, sum = 0 , prod = 1;

  do {
     scanf("%d", &input);
     if (input <=0 )
        break;
     sum = sum + input ;
     printf("partial sum: %d\n", sum);
     prod = prod * input ;
     } while (1) ;

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

return 0;
}
```

Έξοδος από βρόχο Ούτως ή άλλως μόνο με break =>

Απλή συνθήκη στο while

Αρχικοποίηση του input για εξασφάλιση εισόδου στο βρόχο while

έκδοση 5

```
#include <stdio.h>

int main( )
{

  int input=1, sum = 0 , prod = 1;

  while ( input > 0) {
     scanf("%d", &input) ;

     if (input <=0 )
        break;

     sum = sum + input ;
     printf("partial sum: %d\n", sum);
     prod = prod * input ;
     }

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);
  return 0;
}
```

```c
#include <stdio.h>
int getinput(void) ;

int main() {
      int a;
      while ( (a = getinput()) > 0) {
            printf("say something\n");
            printf("%d\n", a);
      }

      return 0;
}

int getinput(void) {
      int a;
      scanf("%d", &a);
      return a;
}
```

Σε τι διαφέρουν οι εκφράσεις

```c
a = getinput() > 0

(a = getinput()) > 0

a = (getinput() > 0)
```

Έκδοση 6

```c
#include <stdio.h>
int getinput(void);

int main( )
{
  int input, sum=0, prod = 1 ;

  while ((input=getinput())>0) {
        sum = sum + input ;
        printf("partial sum: %d\n", sum);
        prod = prod * input;
      }

  printf("sum: %d\n", sum);
  printf("product: %d\n", prod);

  return 0;
}
```

```c
int getinput(void) {
      int a;
      scanf("%d", &a);
      return a;
}
```

έκδοση 7

```c
#include <stdio.h>
int getinput(void);
int partialsums(void);
void report(int);

int main( )
{
  int sum ;

  sum = partialsums();
  report(sum);
  return 0;
}
int getinput(void) {
   int a;
   scanf("%d", &a);
   return a;
}
```

```c
void report(int sum) {
  printf("sum: %d\n", sum);
}

int partialsums(void) {
  int input ;
  int sum = 0 ;
  while ((input=getinput())>0) {
        sum = sum + input ;
        printf("partial sum: %d\n", sum);

   }
  return sum;
}
```

Παράδειγμα – Λεκτική περιγραφή λύσης (2)

- Ζήτησε από το χρήστη **να επιλέξει μεταξύ τριών επιλογών**:
- Ανάλογα με την userchoice
  - αν είναι 1, ξεκίνησε τη διαδικασία, `getchoice( )`
  - αν είναι 2, σταμάτα τη διαδικασία, `start( )`
  - αν είναι 3, να λήξει η εκτέλεση του προγράμματος. `stop( )`
- Συνέχισε να ζητάς επιλογή από χρήστη **έως ότου επιλεγεί η λήξη** του προγράμματος.

Λεκτική περιγραφή λύσης

```
userchoice = getchoice();
while (δεν επιλέχθηκε η λήξη) {
Ανάλογα με την τιμή της userchoice:
  αν είναι 1, start();
  αν είναι 2, stop();
  αν είναι 3, να λήξει η εκτέλεση του
   προγράμματος.
userchoice = getchoice();
}
```

Οργάνωση βασικού βρόχου (1)

```
int main ( ) {
    int userchoice ;

    userchoice = getchoice ( ) ;

    while (userchoice != 3 ) {
            switch (userchoice) {
               case 1: start( ); break;
               case 2: stop( ); break;
               default: break;
            }
             userchoice =  getchoice() ;
    }

    return 0;
}
```

```
int main ( ){
int userchoice ;

userchoice = getchoice();

while (userchoice != 3) {
      switch (userchoice) {
      case 1: start();
break;
      case 2: stop();
break;
      default:break;
          }

      userchoice=getchoice();

      }

return 0;
}
```

```
int main ( ) {
int userchoice ;

while((userchoice=getchoice())!= 3)
   {
      switch (userchoice)
       {
       case 1: start(); break;
       case 2: stop(); break;
       default: break;
       }

   }
return 0;
}
```

```c
#include <stdio.h>
int getchoice (void) ;
void start (void) ;
void stop (void);

int main (){

    int userchoice ;

    while ((userchoice = getchoice()) != 3){
     switch (userchoice) {
        case 1: start() ;
                break;
        case 2: stop();
                break;
        default: break;
     }
   }

   return 0;
}
```

```c
int getchoice (void ) {
 int choice ;

 printf("1: start\n2: stop\n3:quit\n");
 printf("enter choice:\n");
 scanf("%d", &choice);

 return choice;
}

void start (void) {
    printf("Start...");
}

void stop (void) {
    printf("Stop...");
}
```

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

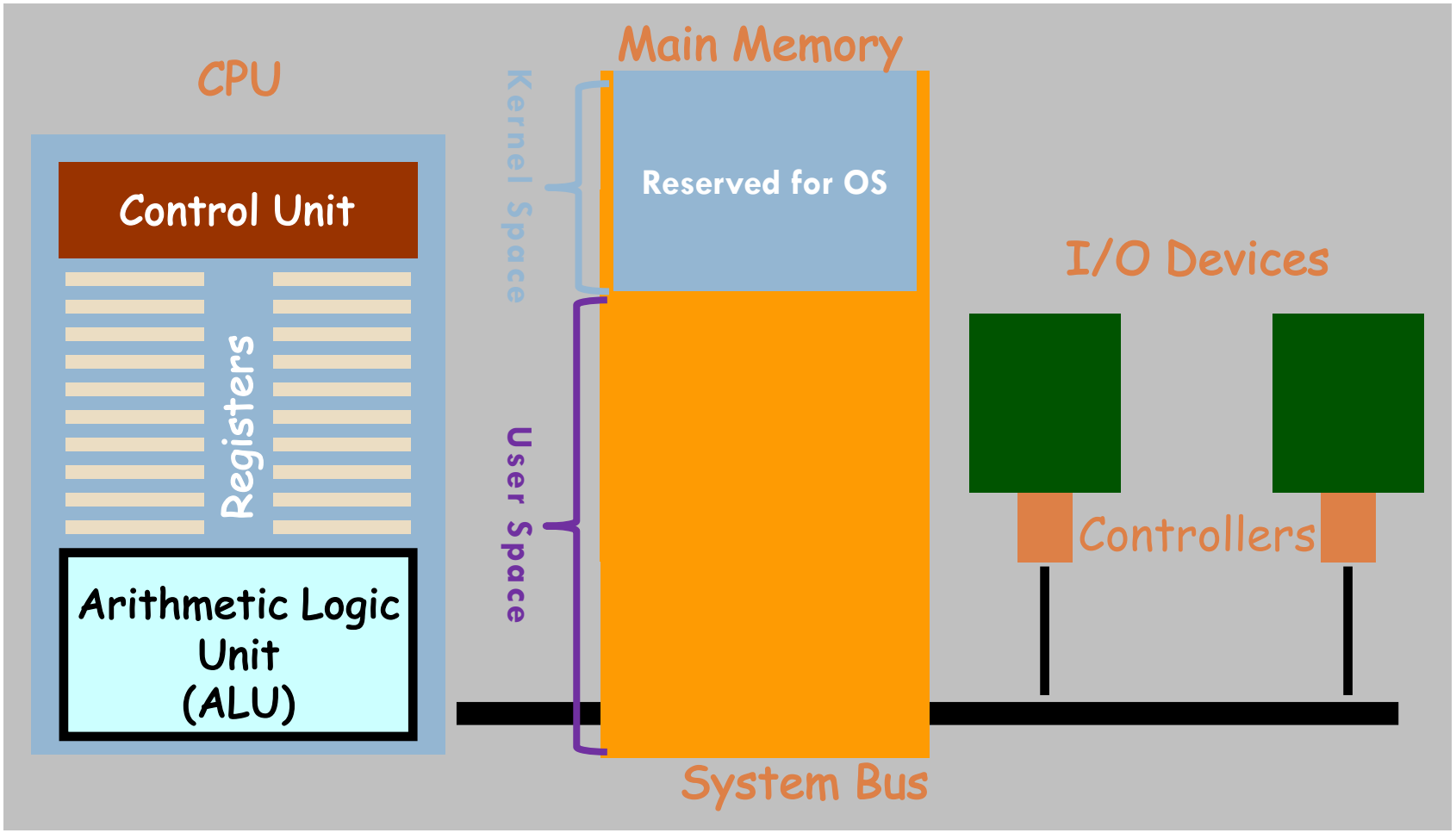
- https://eclass.upatras.gr/

ΔΙΑΔΙΚΑΣΤΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

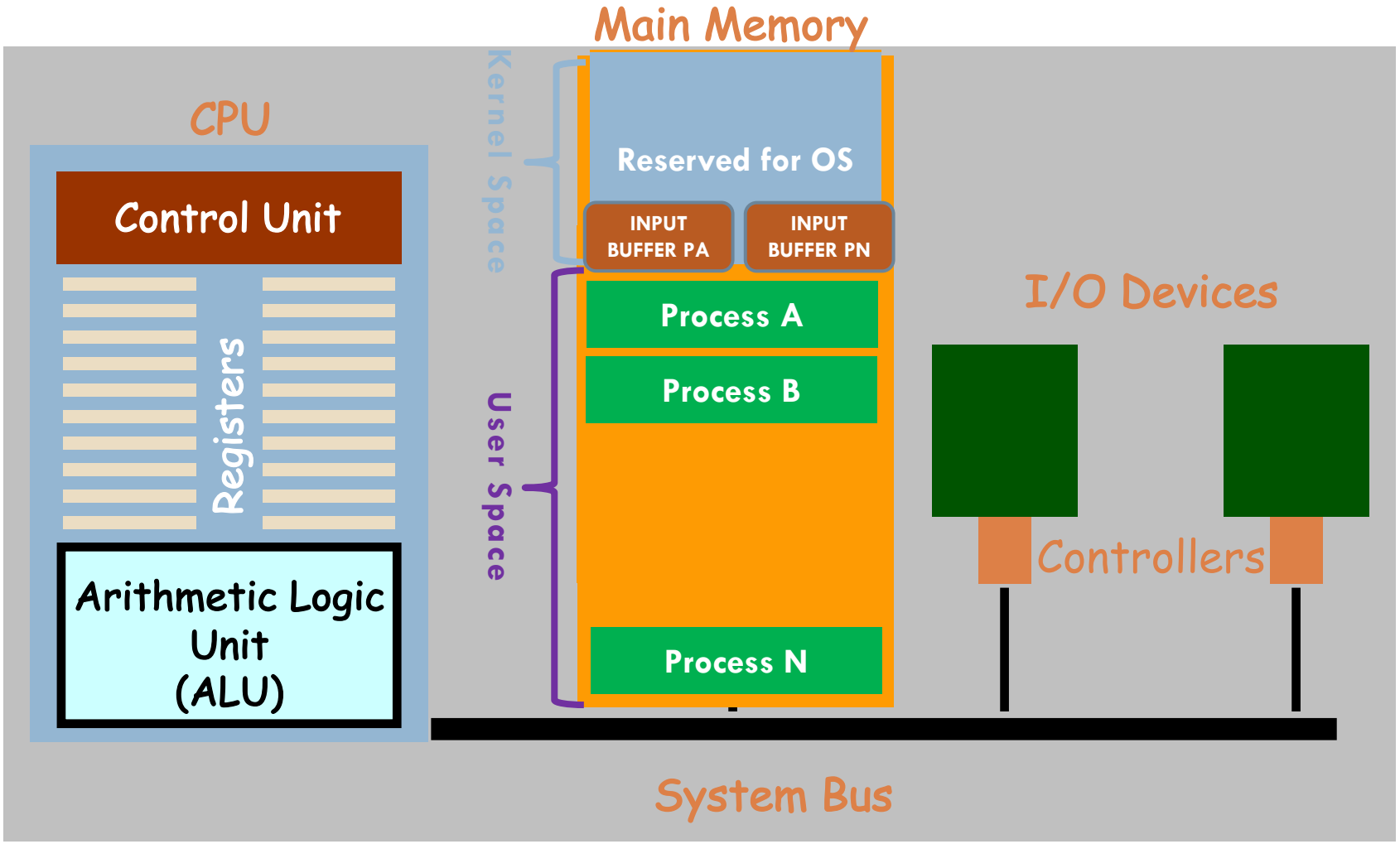
Σε αυτό το μάθημα αναλύουμε απορίες της εβδομάδας

2η Εβδομάδα: Στοιχεία της Γλώσσας C

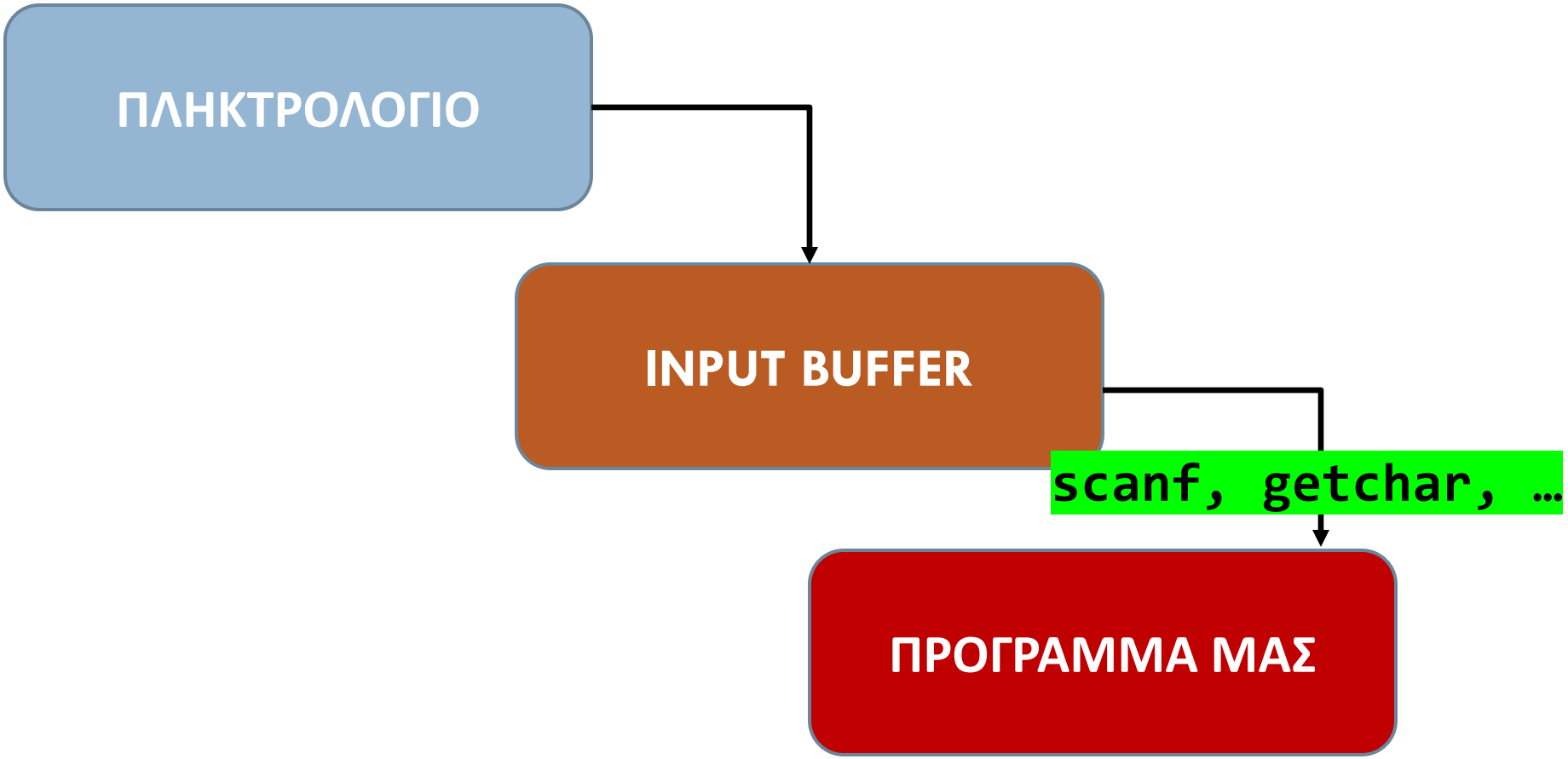
Αρχιτεκτονική Η/Υ



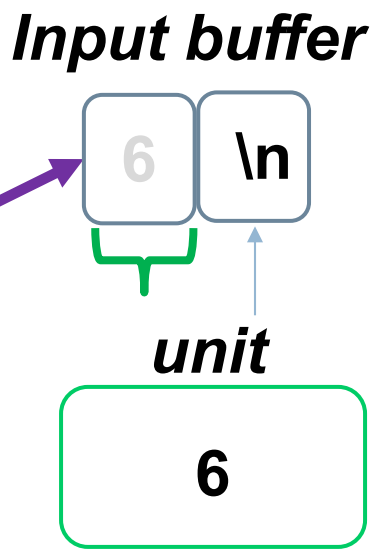
Διασύνδεση μεταξύ των εφαρμογών και της μηχανής



Input Buffer

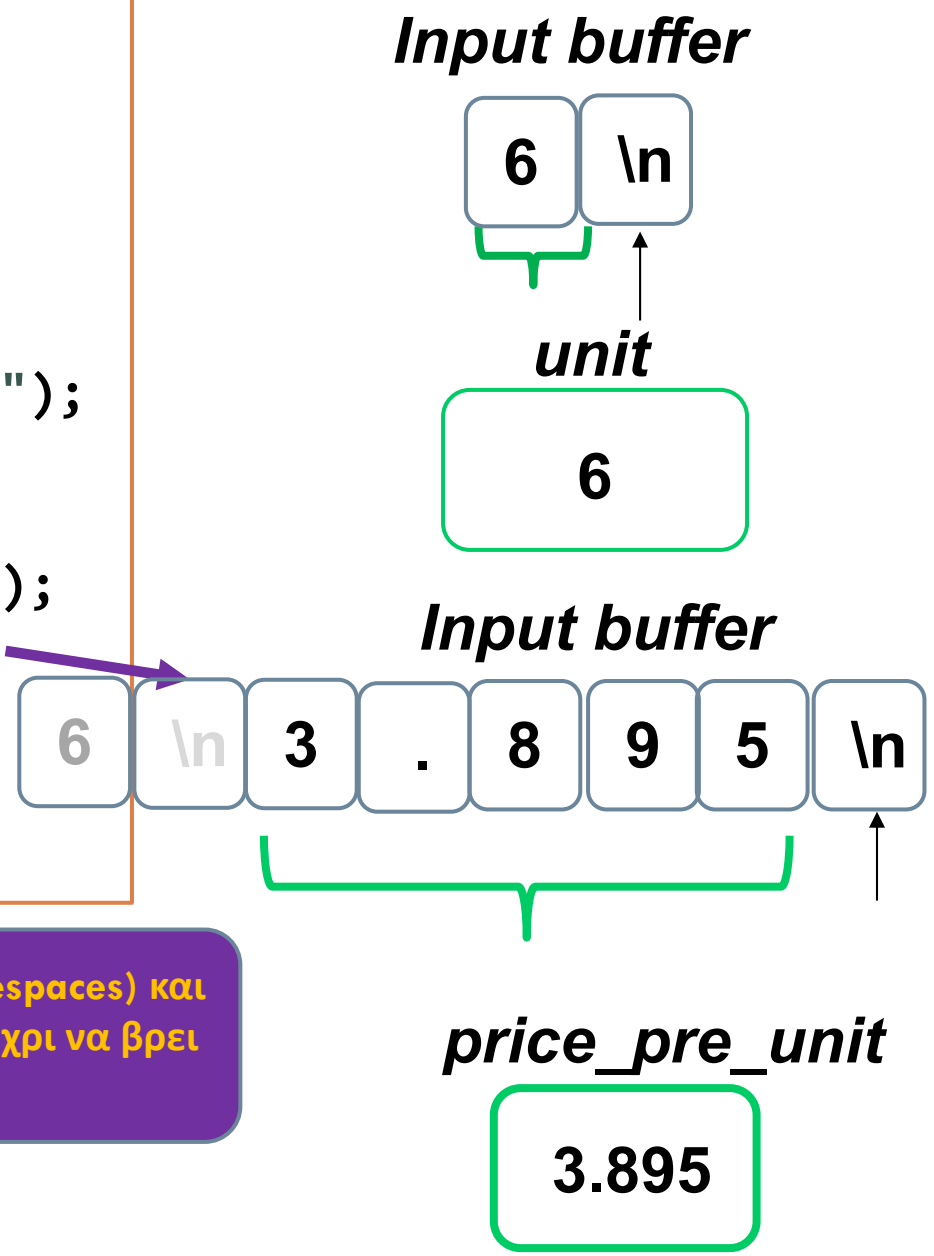


```c
#include <stdio.h>

int main( )
{
  int unit=0;
  double price_per_unit=0.0;

  printf(”Enter number of units");
  scanf(“%d”,&unit);

  printf(”Enter price per unit");
  scanf(“%lf”,&price_per_unit);
  return 0;
}
```



scanf σταματά στο newline και τοποθετεί την τιμή 6 στη θέση μνήμης της μεταβλητής unit

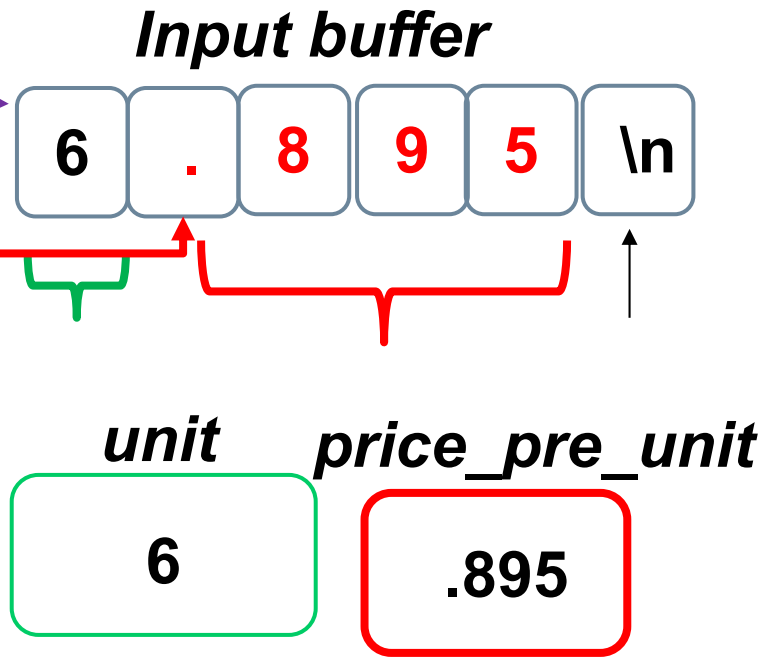
```c
#include <stdio.h>

int main( )
{
  int unit=0;
  double price_per_unit=0.0;

  printf(”Enter number of units");
  scanf(“%d”,&unit);

  printf(”Enter price per unit");
  scanf(“%lf”,&price_per_unit);
  return 0;
}
```



Η δεύτερη scanf αγνοεί το newline (ή άλλα whitespaces) και συνεχίζει να διαβάζει τον πραγματικό αριθμό μέχρι να βρει το επόμενο newline

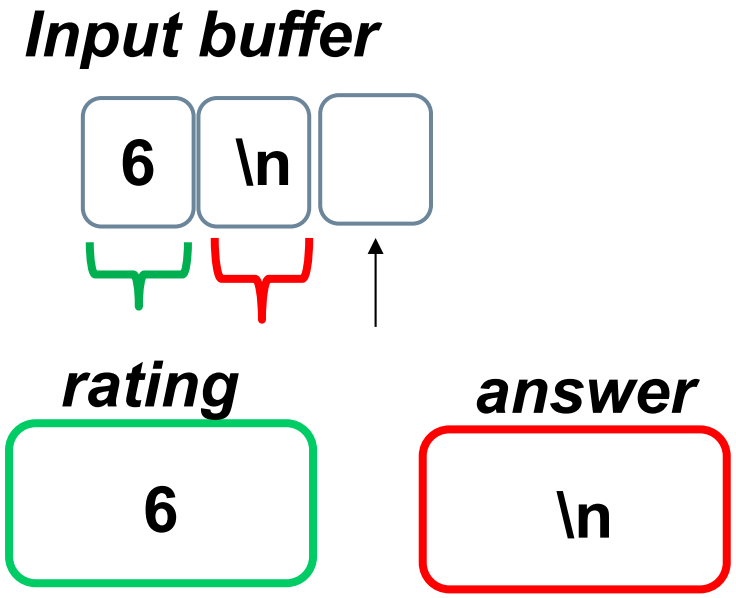
```c
#include <stdio.h>

int main( )
{
  int unit=0;
  double price_per_unit=0.0;

  printf(”Enter number of units");
  scanf(“%d”,&unit);

  printf(”Enter price per unit");
  scanf(“%lf”,&price_per_unit);
  return 0;
}
```

Ερώτηση:
Τι θα γίνει αν ο χρήστης (στην πρώτη scanf) αντί για ακέραιο πληκτρολογήσει πραγματικό αριθμό;

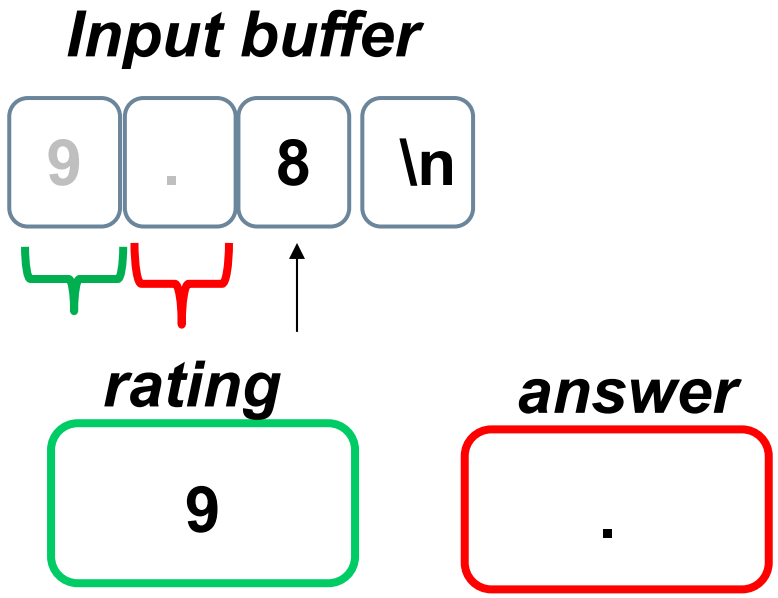


```
Enter rating6
Sure (Y/N)
Process returned 0 (0x0)   execution time : 2.230 s
Press ENTER to continue.
```

```c
#include <stdio.h>

int main( )
{
  int rate_me=0;
  char answer;

  printf(”Enter rating");
  scanf(“%d”,&rate_me);

  printf(”Sure (Y/N)");
  scanf(“%c”,&answer);
  return 0;
}
```

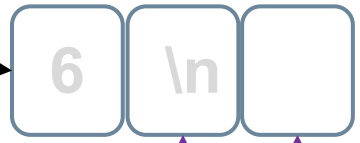


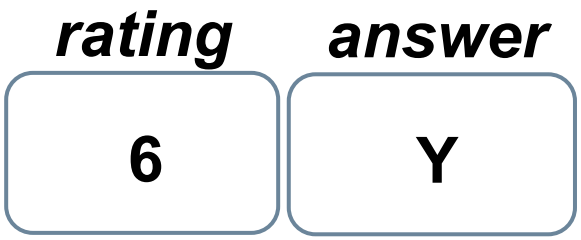
```c
#include <stdio.h>

int main( )
{
  int rate_me=0;
  char answer;

  printf(”Enter rating");
  scanf(“%d”,&rate_me);

  printf(”Sure (Y/N)");
  scanf(“%c”,&answer);
  return 0;
}
```



```c
#include <stdio.h>

int main( )
{
   int rate_me=0;
   char answer;

   printf(”Enter rating");
   scanf(“%d”,&rate_me);

   while(getchar()!=‘\n’);

   printf(”Sure (Y/N)");
   scanf(“%c”,&answer);
   return 0;
}
```

getchar()
Διαβάζει από τον input buffer τα περιεχόμενά του






```
Enter rating10
Sure (Y/N)Y

Process returned 0 (0x0)   execution time : 6.539 s
```

Παραδείγματα


```c
#include <stdio.h>
#include <stdlib.h>

int get_input_buffer_length(void);
int main(void)
{
    int x;
    printf("\n Please an integer:");
    scanf("%d",&x);

    while(get_input_buffer_length()>0){
        printf("\n Please an integer:");
        scanf("%d",&x);
    }
    printf("\nYou entered: %d ", x);
    return 0;
}

int get_input_buffer_length(void){
  int len=0;
  while (getchar() != '\n' ) len++;
  return len;
}
```

Ερώτηση:
Πώς μπορώ να είμαι σίγουρος ότι ο χρήστης πληκτρολόγησε ακέραιο;

```
 Please an integer:ABC

 Please an integer:7.8

 Please an integer:777

You entered: 777
```

Υπάρχει καλύτερη υλοποίηση;

```c
#include <stdlib.h>
float get_float(void);
int get_input_buffer_length(void);
int main(void)
{
    float f=get_float();
    return 0;
}

float get_float(void){
   float x;
   printf("\n Please enter a floating number:");
   scanf("%f",&x);

   while(get_input_buffer_length()>0){
       printf("\n Please enter a floating number:");
       scanf("%f",&x);
   }
   printf("\nYou entered: %f ", x);
   return x;
}

int get_integer(void){
   int x;
   printf("\n Please an integer:");
   scanf("%d",&x);

   while(get_input_buffer_length()>0){
       printf("\n Please an integer:");
       scanf("%d",&x);
   }
   printf("\nYou entered: %d ", x);
   return x;
}


int get_input_buffer_length(void){
  int len=0;
  while (getchar() != '\n' ) len++;
  return len;
}
```

Ερώτηση:
Πώς μπορώ να είμαι σίγουρος ότι ο χρήστης πληκτρολόγησε πραγματικό αριθμό;

```
Please enter a floating number:t

Please enter a floating number:dfgert

Please enter a floating number:7,7

Please enter a floating number:7.7

You entered: 7.700000
```

Υπάρχει καλύτερη υλοποίηση;

Παραδείγματα


```c
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
    char mychar;
    char mychar2;

    printf("Enter a character:");
    scanf("%c",&mychar);

    printf("Enter a second character:");
    scanf("%c",&mychar2);

    printf("The characters you entered are %c and %c",mychar,mychar2);
    return;
}
```

```
Enter a character:A
Enter a second character:
The characters you entered are A and
```

Παραδείγματα

Ερώτηση:
Υπάρχει λύση εκτός από να ελέγχω μόνος μου τον input buffer;


```
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
    char mychar;
    char mychar2;

    printf("Enter a character:");
    scanf("%c",&mychar);

    printf("Enter a second character:");
    scanf("\n%c",&mychar2);

    printf("\nThe characters you entered are %c and %c",mychar,mychar2);
    return;
}
```

```
Enter a character:A
Enter a second character:B

The characters you entered are A and B
```

Αγνόησε το newline και ξεκίνα να διαβάζεις από την επόμενη θέση του input bufffer

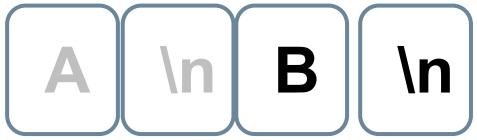
Παραδείγματα

Ερώτηση:
Υπάρχει λύση εκτός από να ελέγχω μόνος μου τον input buffer;


```
Enter a character:A
Enter a second character:B

The characters you entered are A and B
```

```c
#include <stdio.h>
#include <stdlib.h>
void main(void)
{
    char mychar;
    char mychar2;

    printf("Enter a character:");
    scanf("%c",&mychar);

    printf("Enter a second character:");
    scanf("\n%c",&mychar2);

    printf("\nThe characters you entered are %c and %c",mychar,mychar2);
    return;
}
```

Input buffer



Αγνόησε το newline και ξεκίνα να διαβάζεις από την επόμενη θέση του input bufffer

Ευχαριστώ για την προσοχή σας

Επικοινωνία

- Skype: fidas.christos
- Email: fidas@upatras.gr
- Phone: 2610 – 996491
- Web: http://cfidas.info

Άμεση Επικοινωνία μέσω Skype

SkypeID:
fidas.christos

- Ώρες γραφείου: Τετάρτη & Παρασκεύη 11:00-13:00

Join Zoom Meeting
https://upatras-gr.zoom.us/j/95080297961?pwd=MzRta0JRd3ZwVEVrREZNc09qbG1Zdz09

Το υλικό της διάλεξης είναι διαθέσιμο στο eclass

- https://eclass.upatras.gr/