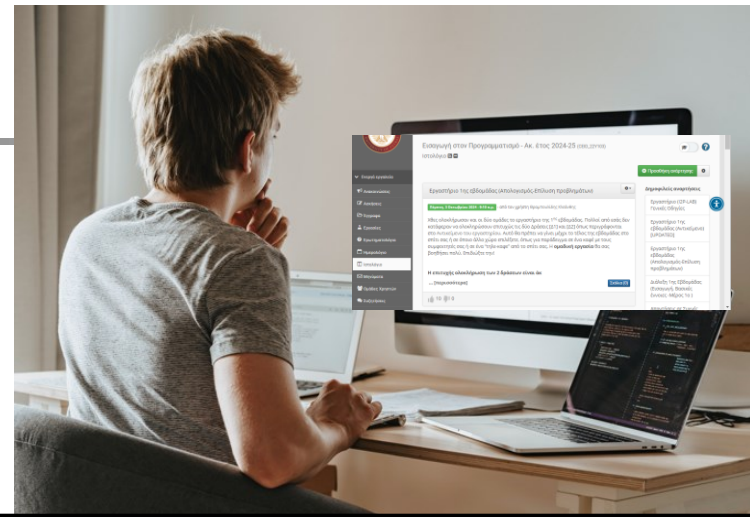
# Εισαγωγή στον Προγραμματισμό

**(CEID_NY131)**



Βασικές έννοιες Διαδικαστικού Προστακτικού Προγραμματισμού (2ο Μέρος )

με τη χρήση παραδειγμάτων Υπολογιστικών διεργασιών

Kleanthis Thramboulidis
Prof. of Software and System Engineering
University of Patras
https://sites.google.com/site/thramboulidiskleanthis/

## Βασικές έννοιες Προστακτικού Προγραμματισμού

Imperative programming

Procedural Abstraction

Data Abstraction

Function

Selection Statement

*Variable*

Statement

*Data Type*

*Loop Statement*

...τίποτε παραπάνω από αντιγραφή βασικών εννοιών από την καθημερινή ζωή

# Προγραμματισμός – Βασικές έννοιες

- Διεργασία – **Συνάρτηση (Function)**
- Αλγόριθμος – Περιγραφή Διεργασίας
- Λεκτική περιγραφή – Ορίζει τον Αλγόριθμο
- Πρόγραμμα (Πηγαίο – Εκτελέσιμο)
- Μεταγλωττιστής (Compiler)
- IDE (Integrated Development Environment)
- Αφαιρετικότητα (Abstraction)
  - Στις διεργασίες (procedural abstraction)
  - Στα δεδομένα (data abstraction)

**Αλγόριθμος:** ένα σύνολο βημάτων τα οποία καθορίζουν τον τρόπο εκτέλεσης μίας εργασίας.



# Οργάνωση Διάλεξης - Παραγωγή Πηγαίου Κώδικα

- **Παραγωγή Πηγαίου Κώδικα**
  - Λεκτική Περιγραφή σε Πηγαίο Κώδικα (Add2Numbers Version1)
  - How I feel when my code works
- Αφαιρετικότητα στις Διεργασίες
  - Ορισμός – Είδη - Στοιχεία Διεργασίας
  - Αξιοποίηση Αφαιρετικότητας στον προγραμματισμό (Add2Numbers Version 2)
  - Η συνάρτηση sum (Add2Numbers Version 3)
- Ανάθεση έργου - Παραδείγματα
  - Η Hailstone sequence (Λεκτική περιγραφή)

# Add2Numbers – Step2:πηγαίος κώδικας 1/2

Ζήτα τον πρώτο αριθμό από τον χρήστη

**printf( … );** // **συνάρτηση** (function) **βασικής**
// **βιβλιοθήκης**

**Πάρε** τον πρώτο αριθμό από τον χρήστη

**scanf(…);** // **συνάρτηση βασικής βιβλιοθήκης**

Γράψτε, μεταγλωττίστε και εκτελέστε το πρόγραμμα που διαβάζει τον πρώτο αριθμό.

Ελέγξτε αν το σύστημα πήρε και αποθήκευσε σωστά τον αριθμό που έδωσε ο χρήστης.

**Αυξητική Ανάπτυξη**

Τα βασικά για συναρτήσεις βασικής βιβλιοθήκης θα βρείτε στο I2P_Functions (slides 4-8)



# Η scanf και η ανάγκη δήλωσης μεταβλητής

```
6   int num1;    // protasi dilwsis genikis (global) me
7
8   int main(int argc, char *argv[]) {
9       printf("Programa prosthesis 2 arithmwn \n"); /
10      printf("Dose ton 1o arithmo:");
11      scanf("%d",&num1);  //klisi synartisis scanf
```

Η **scanf διαβάζει δεδομένα** (data) από τη βασική είσοδο/stdin (πληκτρολόγιο) και τα αποθηκεύει εκεί που θα της ζητηθεί (**&num1**).

**Προϋποθέσεις κλήσης της scanf**:
1. Να έχει δεσμευτεί χώρος στην μνήμη όπου θα αποθηκεύσει αυτό που θα διαβάσει (line 6),
2. Να έχει ανατεθεί στο χώρο αυτό ένα όνομα (**num1** - line 6),
3. Να έχει δηλωθεί ο τύπος της μεταβλητής (**int** - line 6),
4. Να ενημερωθεί για το πως θα διαβάσει τα δεδομένα ("**%d**" – να διαβάσει την είσοδο ως ακέραιο).

# Πρόταση Δήλωσης μεταβλητής

```
5
6   int num1;    // protasi dilwsis genikis (global) metavlitis
```

Η γραμμή 6 είναι μια πρόταση δήλωσης γενικής (global) μεταβλητής.
Ζητά από το σύστημα να δεσμεύσει χώρο στη μνήμη για την αποθήκευση ενός ακέραιου αριθμού (**int**) και αποδίδει στη θέση αυτή το συμβολικό όνομα **num1** (όνομα μεταβλητής).

- **Μεταβλητή είναι ένα Συμβολικό όνομα**
  - που δίνει λύση στο πρόβλημα της αναφοράς σε πληροφορία της κύριας μνήμης του υπολογιστή.
- Το **όνομα της μεταβλητής** μας επιτρέπει να αναφερόμαστε
  - στη τιμή, χωρίς να είναι απαραίτητο να γνωρίζουμε την ακριβή διεύθυνση της μνήμης που αυτή είναι αποθηκευμένη,
  - στην θέση της μνήμης για να τροποποιήσουμε την τρέχουσα-τιμή της.

count = count + 1

Τα βασικά για μεταβλητές θα βρείτε στο I2P_Variables&DataTypes.pdf (slides 3-8)



# Δήλωση του τύπου της μεταβλητής

```
5
6   int num1;    // protasi dilwsis genikis (global) metavlitis
```

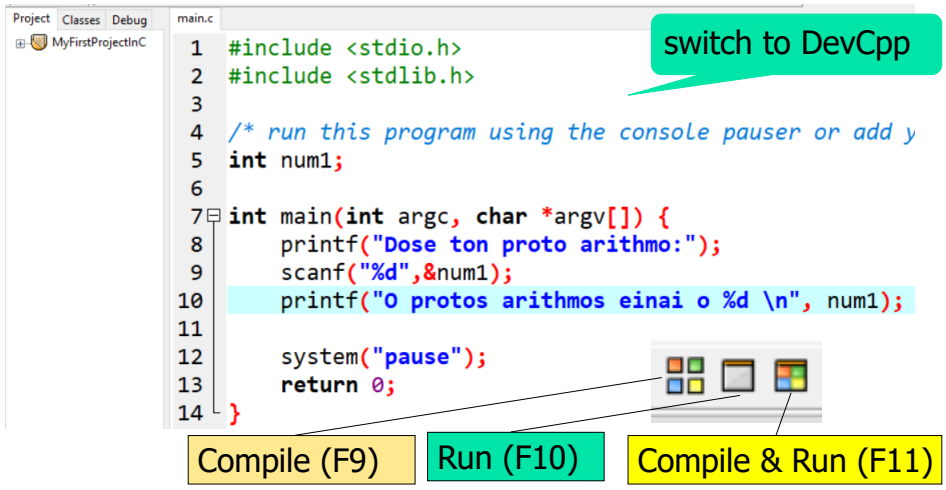
Αν θέλουμε να αποθηκεύσουμε δεδομένα πρέπει **πρώτα να δηλώσουμε** την κατάλληλη μεταβλητή.
Η δήλωση μεταβλητής **προσδιορίζει** εκτός από το όνομα της **και τον τύπο του δεδομένου** για το οποίο αυτή δηλώνεται.

- Οι σύγχρονες προστακτικές γλώσσες
  - περιλαμβάνουν ορισμένους **ενσωματωμένους τύπους**,
    - **int** (ακέραιος), **char** (χαρακτήρας), **float, double …**
  - προσφέρουν ένα **μηχανισμό για τον ορισμό νέων τύπων**
    - struct student { … };
    - struct rectangle { … };

Τα βασικά για τύπους δεδομένων θα βρείτε στο I2P_Variables&DataTypes.pdf (slides 9-14)

# Incremental development

- Αναπτύξτε το πρόγραμμα σας σε διαδοχικούς κύκλους συγγραφής κώδικα, μεταγλώττισης, εκτέλεσης και ελέγχου καλής λειτουργίας (**Αυξητική Ανάπτυξη**)






# Δομή C Προγράμματος

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  /* run this program using the console pauser or add your own getch, system(
5
6  int num1;   // protasi dilwsis genikis (global) metavlitis
7
8  int main(int argc, char *argv[]) {
9      printf("Programa prosthesis 2 arithmwn \n"); //klisi synartisis printf
10     printf("Dose ton 1o arithmo:");
11     scanf("%d",&num1);  //klisi synartisis scanf
12     printf("o 1os arithmos einai: %d\n",num1);
13     return 0;
14 }
```

- Αποτελείται από
  - **Ορισμούς Συναρτήσεων** (Ορισμός συνάρτησης main lines 8-14),
  - Δηλώσεις **μεταβλητών** (Δήλωση μεταβλητής num1 line 6) **και συναρτήσεων**
  - **Ορισμούς τύπων δεδομένων,**
  - Προτάσεις προεπεξεργαστή (lines 1-2),
  - Σχόλια (line 4, part of lines 6,9,11)

# Add2Numbers – Step2:πηγαίος κώδικας 2/2

....

Πρόσθεσε τους δύο αριθμούς

**num1 + num2** // **τελεστής (operator) +**

Αποθήκευσε το αποτέλεσμα

**res = .... ;** // **τελεστής =**

Εμφάνισε το αποτέλεσμα της άθροισης

**printf( ... );** // συνάρτηση βασικής βιβλιοθήκης

Γράψτε, μεταγλωττίστε και εκτελέστε το πρόγραμμα

Ελέγξτε αν το σύστημα εκτελεί σωστά το έργο

Αυξητική Ανάπτυξη



# Τελεστές και Εκφράσεις

```
16      res = num1 + num2;
17      printf("To athrisma einai: %d\n",res);
```

**+** : Ο **τελεστής** + αναπαριστά τη διεργασία της πρόσθεσης αριθμών. Δρα πάνω σε 2 αριθμούς (num1 και num2) και έχει ως αποτέλεσμα το άθροισμα τους. Το αποτέλεσμα της άθροισης είναι η **τιμή της έκφρασης** (num1+num2) την οποία σχηματίζει ο τελεστής με τους 2 αριθμούς.

**Τελεστής**

- **είναι**
  - ένα **σύμβολο** ή μια **λέξη** της γλώσσας προγραμματισμού.
- **αναπαριστά**
  - συγκεκριμένη διεργασία, που εκτελείται πάνω σε ένα ή περισσότερα δεδομένα (τελεστέους - operands)
- **χρησιμοποιείται**
  - για τον σχηματισμό εκφράσεων (expressions).

Τα βασικά για τελεστές και εκφράσεις θα βρείτε στο I2P_Expressions&Operators (slides 2-7)

# Add2Numbers -Πηγαίος κώδικας V1

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

/* run this program using the console pauser or add your ow

int num1,num2;  // protasi dilwsis genikis (global) metavli
int res;

int main(int argc, char *argv[]) {
    printf("Programa prosthesis 2 arithmwn \n"); //klisi sy
    printf("Dose ton 1o arithmo:");
    scanf("%d",&num1);  //klisi synartisis scanf
    //printf("o 1os arithmos einai: %d\n",num1);
    printf("To athrisma einai: %d\n",res);
    return 0;
}
```

Standard C library
https://www.csse.uwa.edu.au/programming/ansic-library.html
Standard C Library Functions Table, By Name - IBM
https://www.ibm.com/docs/en/i/7.3?topic=extensions-standard-c-library-functions-t

switch to DevCpp

```
C:\Code\courses\I2P24-25\ca
Programa prosthesis 2 arithmwn
Dose ton 1o arithmo:12
Dose ton 2o arithmo:24
To athrisma einai: 36

--------------------------------
Process exited after 7.395 seconds with return value 0
Press any key to continue . . .
```



# DevC++ και η πρόταση system("pause");

**Άσκηση**

- Δημιουργείστε το εκτελέσιμο πρόγραμμα add2Numbers της έκδοσης με την πρόταση system("pause");
- Δημιουργείστε το εκτελέσιμο πρόγραμμα add2Numbers της έκδοσης χωρίς την πρόταση system("pause");
- Τρέξτε τα δυο εκτελέσιμα
  - Α) μέσα από το περιβάλλον του DevC++
  - Β) έξω από το περιβάλλον του DevC++
- Εντοπίστε τη διαφορά στην συμπεριφορά των δύο προγραμμάτων στα δυο διαφορετικά περιβάλλοντα εκτέλεσης, δηλαδή τα DevC++ και το Λειτουργικό Σύστημα (Operating System).

## How I feel when my code works




## Οργάνωση Διάλεξης - Αφαιρετικότητα στις Διεργασίες

- Παραγωγή Πηγαίου Κώδικα
  - Λεκτική Περιγραφή σε Πηγαίο Κώδικα (Add2Numbers Version1)
  - How I feel when my code works
- **Αφαιρετικότητα στις Διεργασίες**
  - Ορισμός – Είδη - Στοιχεία Διεργασίας
  - Αξιοποίηση Αφαιρετικότητας στον προγραμματισμό (Add2Numbers Version 2)
  - Η συνάρτηση sum (Add2Numbers Version 3)
- Ανάθεση έργου - Παραδείγματα
  - Η Hailstone sequence (Λεκτική περιγραφή)

# Αφαιρετικότητα

- μια απλοποιημένη περιγραφή ή τεκμηρίωση που δίνει έμφαση σε ορισμένα χαρακτηριστικά ενώ ταυτόχρονα αποσιώπα άλλα.
- μια από τις βασικότερες τεχνικές με την οποία ο άνθρωπος αντιμετωπίζει την πολυπλοκότητα.

## Είδη αφαιρετικότητας

- αφαιρετικότητα στις διεργασίες (Procedural abstraction)
  - "φτιάξε ένα καφέ"
  - «πάρε με τηλέφωνο»
- αφαιρετικότητα στα δεδομένα (data abstraction)
  - Δώσε μου ένα "βιογραφικό"
  - Δώσε μου την "αναλυτική καρτέλα σου"



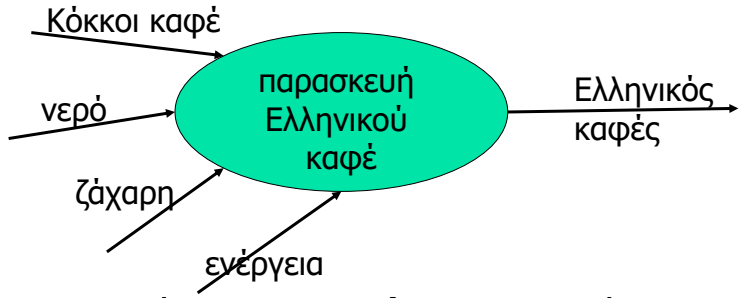
# Αφαιρετικότητα στις διεργασίες

- επιτρέπει στο χρήστη να **εστιάσει τη προσοχή** του
  - στο **τι** γίνεται

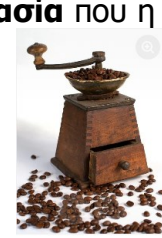
  και όχι
  - στο **πώς** αυτό γίνεται
- επιτυγχάνεται ορίζοντας δύο τμήματα για μι διεργασία
  - τον τρόπο αναφοράς στην διεργασία - **interfac**
  - την υλοποίηση της διεργασίας - **implementation**

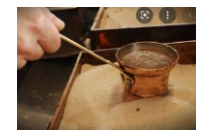

# Η διεργασία παρασκευής καφέ

*Ένα Ελληνικό μέτριο παρακαλώ*




- Είναι μια **σύνθετη** (παραγόμενη) **διεργασία** που η υλοποίηση της δίνεται με την χρήση άλλων διεργασιών
  - Άλεσε τους κόκκους του καφέ
  - Ανάμειξε τα συνθετικά του καφέ
  - Βράσε το μείγμα
  - ...







# Η διεργασία παρασκευής καφέ (συνθετικά)


- Η διεργασία έχει :
  - **Όνομα** ( Ορίζει το **τι κάνει** η διεργασία)
  - **Περιγραφή** (Ορίζει το **πως το κάνει**)
  - Δέχεται **εισόδους** (τις οποίες επεξεργάζεται)
  - Παράγει **έξοδο/δους** (τις οποίες διαμορφώνει με βάση τις εισόδους)

# Παράδειγμα εφαρμογής Αφαιρετικότητας (Add2Numbers)

Add2Numbres –Λεκτική περιγραφή 1η εκδοχή

1. Ζήτα τον πρώτο αριθμό από τον χρήστη → printf(…)
2. Πάρε τον πρώτο αριθμό από τον χρήστη → scanf(…)
3. Ζήτα τον δεύτερο αριθμό από τον χρήστη
4. Πάρε τον δεύτερο αριθμό από τον χρήστη
5. Υπολόγισε το άθροισμα των δύο αριθμών
6. Εμφάνισε το άθροισμα

Add2Numbres –Λεκτική περιγραφή 2η εκδοχή

1. Πάρε τον πρώτο αριθμό από τον χρήστη → getNumber(…)
2. Πάρε τον δεύτερο αριθμό από τον χρήστη
3. Υπολόγισε το άθροισμα των δύο αριθμών
4. Εμφάνισε το άθροισμα

Συνάρτηση ορισμένη από τον προγραμματιστή



# Πηγαίος κώδικας με την getNumber()

Add2Numbres –Λεκτική περιγραφή 2η εκδοχή

1. Πάρε τον πρώτο αριθμό από τον χρήστη → getNumber(…)
2. Πάρε τον δεύτερο αριθμό από τον χρήστη → getNumber(…)
3. Υπολόγισε το άθροισμα των δύο αριθμών → num1 + num2;
4. Εμφάνισε το άθροισμα → printf(…)

Η getNumber **πρέπει να ζητάει** από τον χρήστη ένα αριθμό, **να** τον **διαβάζει** και **να** τον **επιστρέφει** στην συνάρτηση που την κάλεσε.

**Προϋποθέσεις χρήσης**:

1. Να έχει **δηλωθεί** η συνάρτηση
   Αυτό γίνεται με μια **πρόταση δήλωσης συνάρτησης**, η οποία ορίζει το όνομα της, τα ορίσματα που δέχεται και την επιστρεφόμενη τιμή.
2. Να έχει **οριστεί** η συνάρτηση
   Αυτό γίνεται ορίζοντας το **σώμα της συνάρτησης** με τρόπο που να a) αξιοποιεί τα ορίσματα που δέχεται, b) να εκτελεί την διεργασία για την οποία δημιουργείται και c) να επιστρέφει το αποτέλεσμα της διεργασίας.

# getNumber() - Η δήλωση συνάρτησης

```
6   int getNumber(void);  // protasi dilwsis Synartisis
7
8   int num1,num2;  // protasi dilwsis genikis (global) me
9   int res;
```

- Η δήλωση της συνάρτησης
  - Ορίζει
    - Το όνομα της συνάρτησης (getNumber)
    - Τα ονόματα και τους τύπους των ορισμάτων που αυτή δέχεται (void)
    - Τον τύπο της επιστρεφόμενης τιμής (int)
  - τοποθετείται πριν τη δήλωση των γενικών μεταβλητών
  - είναι απαραίτητη για την κλήση της συνάρτησης και τον έλεγχο τύπων (type checking)

**void**: η λέξη σημαίνει πως η συνάρτηση δεν δέχεται ορίσματα



# getNumber() - Ο ορισμός της συνάρτησης

```
21  int getNumber(void){        // Orismos Synartisis
22      int num;
23      printf("Dose arithmo:");
24      scanf("%d",&num);  //klisi synartisis scanf
25      return num;  //epistrefei tin timi tis num
26  }
```

Ο ορισμός της συνάρτησης

- **Ορίζει** το σώμα της συνάρτησης (lines 22-25)
  - Αυτό πρέπει να υλοποιεί τη διεργασία που το όνομα της δηλώνει. Πρέπει να ζητάει από τον χρήστη ένα αριθμό (με την printf -line 23), να τον διαβάζει (με την scanf -line 24) και τον επιστρέφει (με την πρόταση return -line 25) στην συνάρτηση που την κάλεσε (η οποία στην περίπτωση μας είναι η main).
- **Επιστρέφει** τιμή, του τύπου που ορίζει η δήλωση της συνάρτησης (**int**)
- Η 1η πρόταση ορισμού της συνάρτησης (line 21) πρέπει να είναι <u>σύμφωνη με την δήλωση της</u>.

# getNumber() - Η κλήση της συνάρτησης

```
13      num1 = getNumber();   // klisi synartisis kai apodosi epistrefomenis
14      //printf("o 1os arithmos einai: %d\n",num1);
15      num2 = getNumber();
```

## Η κλήση της συνάρτησης

- Γίνεται με το όνομα της ακολουθούμενο από τις τιμές που περνάμε στα ορίσματα της (getNumber())
- Αν επιστρέφει τιμή αυτή πρέπει να ανατεθεί σε μία μεταβλητή (num1 = getNumber()) ή να αξιοποιηθεί στο σχηματισμό μιας έκφρασης, π.χ.

  getNumber() + getNumber())

Ο compiler ελέγχει αν η κλήση γίνεται σύμφωνα με την δήλωση της (έλεγχος τύπων)

Δοκιμάστε την πρόταση

```
res = getNumber() + getNumber();
```



# Η συνάρτηση sum() (εκδοχή 1η)

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   void sum(void); // protasi dilwsis synartisis
5
6   int num1,num2;  // protasi dilwsis genikis (global) metavlitis
7   int result;
8
9   int main(int argc, char *argv[]) {
10      printf("Programa prosthesis 2 arithmwn me synartisi (V1)\n");
11      printf("Dose ton 1o arithmo:");
12      scanf("%d",&num1);
13      //printf("o 1os arithmos einai: %d\n",num1);
14      printf("Dose ton 2o arithmo:");
15      scanf("%d",&num2);
16      //printf("o 2os arithmos einai: %d\n",num2);
17      sum();    // protasi klisis synartisis
18      printf("to athrisma twn %d kai %d einai: %d\n",num1,num2,result);
19      return 0;
20  }
21
22  void sum(void){         //orismos synartisis
23      result=num1+num2;
24  }
```

Δήλωση συνάρτησης

Κλήση συνάρτησης

Ορισμός συνάρτησης

Η συνάρτηση sum **χρησιμοποιεί** τις **γενικές μεταβλητές** num1, num2 και result οπότε **δεν είναι** επαναχρησιμοποιήσιμη (**reusable**)

# Η συνάρτηση sum() (εκδοχή 2η)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int sum(int n1, int n2); // protasi dilwsis synartisis
5  //int num1,num2;     // protasi dilwsis genikis (global) metavlitis
6  //int result;
7  int main(int argc, char *argv[]) {
8      int num1,num2; // protasi dilwsis topikis (local) metavlitis
9      int result;
10
11     printf("Programa prosthesis 2 arithmwn me synartisi (V1)\n");
12     printf("Dose ton 1o arithmo:");
13     scanf("%d",&num1);
14     printf("Dose ton 2o arithmo:");
15     scanf("%d",&num2);
16     result = sum(num1,num2);          // protasi klisis synartisis
17     printf("to athrisma twn %d kai %d einai: %d\n",num1,num2,result);
18     return 0;
19 }
20
21 int sum(int n1, int n2){          //orismos synartisis
22     int result;
23     result=n1+n2;
24     return result;
25 }
```

Δήλωση συνάρτησης

Τοπικές μεταβλητές

Κλήση συνάρτησης

Ορισμός συνάρτησης

Η συνάρτηση sum **δεν χρησιμοποιεί γενικές μεταβλητές** οπότε **είναι** επαναχρησιμοποιήσιμη (**reusable**)



# Η συνάρτηση sum() (εκδοχή 2η)

**Δήλωση συνάρτησης**

```
4   int sum(int n1, int n2);
```

**Κλήση συνάρτησης**

```
16      result = sum(num1,num2);
```

**Ορισμός συνάρτησης**

```
21  int sum(int n1, int n2){
22      int result;
23      result=n1+n2;
24      return result;
25  }
```

Τα βασικά για συναρτήσεις βασικής βιβλιοθήκης θα βρείτε στο I2P_Functions (slides 4-8)

# getNumber() – Δύο προτάσεις βελτίωσης

- **Βελτίωση που αφορά το όνομα της συνάρτησης**
  - Καθώς υπάρχουν και αριθμοί κινητής υποδιαστολής απλής ακρίβειας (float) και αριθμοί κινητής υποδιαστολής διπλής ακρίβειας (double) μπορούμε (και είναι καλύτερο) να δώσουμε ως όνομα στην getNumber() το **getInteger** ή απλά **getInt**. Αυτό θα μας επιτρέψει να ορίσουμε άλλες δύο συναρτήσεις τις getFloat() και getDouble().
- **Βελτίωση που αφορά τη λειτουργικότητα**
  - Μια καλύτερη εκδοχή της συνάρτησης είναι να έχουμε τη δυνατότητα να την καλούμε και να μην εμφανίζει πάντα την φράση "Dose arithmo:", ή ότι εσείς έχετε βάλει μέσα στην printf στο σώμα της, αλλά μια διαφορετική φράση την οποία εσείς θα ορίζετε όταν την καλείτε. Αυτό μας δίνει μια πιο επαναχρησιμοποιήσιμη συνάρτηση. Για να το πετύχουμε αυτό θα πρέπει

α) να δηλώσουμε τη συνάρτηση όπως παρακάτω, και

**int getInt(char message[]);** // Τη σημασία θα τη δούμε αργότερα όταν μιλήσουμε για πίνακες (arrays)

Β) μέσα στο σώμα της να αντικαταστήσουμε την printf("Dose arithmo:"); με την printf(message);

Με βάση αυτό η printf θα εμφανίσει ότι περάσουμε ως όρισμα στην κλήση της συνάρτησης.

```
11      num1 = getInt("Dose ton 1o akeraio:");
12      num2 = getInt("Dose ton 2o akeraio:");
```



# Οργάνωση Διάλεξης - Ανάθεση έργου, Παραδείγματα

- Παραγωγή Πηγαίου Κώδικα
  - Λεκτική Περιγραφή σε Πηγαίο Κώδικα (Add2Numbers Version1)
  - How I feel when my code works
- Αφαιρετικότητα στις Διεργασίες
  - Ορισμός – Είδη - Στοιχεία Διεργασίας
  - Αξιοποίηση Αφαιρετικότητας στον προγραμματισμό (Add2Numbers Version 2)
  - Η συνάρτηση sum (Add2Numbers Version 3)
- **Ανάθεση έργου - Παραδείγματα**
  - Η Hailstone sequence (Λεκτική περιγραφή)

# Ανάθεση έργου (hailstone sequence)




# Hailstone sequence

**The Collatz problem:** *one of the world's unsolved problems in mathematics.*
*It involves a sequence of numbers called a Hailstone sequence. It is called this because the numbers go up and down again.*
*The conjecture is that* ***these sequences always reach 1****, no matter which positive integer is chosen to start the sequence.*

Hailstone Sequences **follow these rules**:

- If a number is even, divide it by 2
- If a number is odd, multiply it by 3 and add 1

For 25: 25 – 76 – 38 – 19 – 58 – 29 – 88 – 44 …

# Hailstone sequence -Λεκτική περιγραφή Version 1

- Το έργο

  Γράψε μας την **hailstone sequence** για τον αριθμό 25

- Ανάθεση σε άνθρωπο. Πώς;

*A1. Πάρε έναν θετικό ακέραιο αριθμό*
*A2. Εάν ο αριθμός είναι 1*
*τερμάτισε*
*A3. Εάν ο αριθμός είναι άρτιος*
*τότε διαίρεσε τον με το 2*
*διαφορετικά*
*πολλαπλασίασε τον με το 3 και*
*πρόσθεσε 1*
*A4. Τύπωσε τον αριθμό*
*A5. Επανέλαβε από το βήμα A2.*

**1η εκδοχή**



# Hailstone sequence - Ψευδοκώδικας

- Δεν αποτελεί λεκτική περιγραφή!!!

***A1. [Input n] Πάρε έναν ακέραιο αριθμό***
***A2. [Termination] If n = 1***
***exit***
***A3. [Check n] If n is even***
***n := n / 2***
***else***
***n := n * 3 + 1***
***A4. [Print n]Τύπωσε τον n***
***A5. [Next number] Go to A2.***

Η χρήση του **go to** καταστρέφει τη δόμηση του κώδικα

# Hailstone sequence -Λεκτική περιγραφή Version 2

*A.1. Ζήτα ένα θετικό ακέραιο αριθμό*
*A.2 Πάρε τον αριθμό*
*A.3 **Επανέλαβε** τα παρακάτω*
*A3.1 **Αν** ο αριθμός είναι 1*
*τερμάτισε*
*A3.2 **Αν** ο αριθμός είναι άρτιος*
*Διαίρεσε τον με το 2*
***Διαφορετικά***
*Πολλαπλασίασε τον με το 3 και πρόσθεσε το 1*
*A3.3 Τύπωσε τον αριθμό*

A.3:**Loop Statement**
A3.1: **Decision Statement**
A3.2: Selection Statement

2η εκδοχή

Η πρόταση επανάληψης δεν έχει συνθήκη τερματισμού



# Hailstone sequence -Λεκτική περιγραφή Version 3

*A.1. Ζήτα ένα θετικό ακέραιο αριθμό*
*A.2 Πάρε τον αριθμό*
*A.3 **Επανέλαβε μέχρι** ο αριθμός να γίνει 1*
*A3.1 **Αν** ο αριθμός είναι άρτιος*
*Διαίρεσε τον με το 2*
***Διαφορετικά***
*Πολλαπλασίασε τον με το 3 και πρόσθεσε το 1*
*A3.2 Τύπωσε τον αριθμό*

3η εκδοχή

# How I feel when my code works