242 - Εισαγωγή στους Η/Υ

Τμήμα Μαθηματικών,
Πανεπιστήμιο Ιωαννίνων

Άρτια Α.Μ. (0-2-4-6-8)

Συναρτήσεις

Συναρτήσεις

- Συναρτήσεις
  - Έτοιμος κώδικας για συγκεκριμένους υπολογισμούς στην C/C++
  - Προγράμματα χρησιμοποιούν συναρτήσεις οριζόμενες από χρήστη και συναρτήσεις από τις βιβλιοθήκες της C/C++
    - C/C++ βιβλιοθήκες έχουν πολλές συναρτήσεις για ευρύ φάσμα υπολογισμών και εφαρμογών
- Δήλωση συνάρτησης
  - Όνομα
  - Παραμέτρους
  - Τύπο επιστρεφόμενης τιμής
  - Σώμα: τοπικές μεταβλητές, εκτελούμενες εντολές
- Κλήση συναρτήσεων
  - Γίνεται κατά την χρήση της συνάρτησης
    - Δίνουμε όνομα και παραμέτρους (δεδομένα)
    - Η συνάρτηση εκτελεί υπολογισμούς ή άλλους χειρισμούς
    - Επιστέφει αποτελέσμα

Συναρτήσεις

- Βιβλιοθήκη Math
  - Κοινές μαθηματικές συναρτήσεις
  - **`#include <cmath>`**
- Συντακτικό κλήσης
  - **`FunctionName(`** *arg1, arg2, …, argn***`);`**
  - **`sqrt( 900.0 );`**
    - Καλεί την συνάρτηση υπολογισμού της τετραγωνικής ρίζας ενός πραγματικού
    - Όλες οι συναρτήσεις της βιβλιοθήκης math επιστέφουν τύπο **`double`**
  - Οι παράμετροι της cmath μπορεί να είναι σταθερές μεταβλητές η εκφράσεις.

Συναρτήσεις

- Οι συναρτήσεις
  - Βοηθούν στην καλύτερη δόμηση του προγράμματος – το εκλεπτύνουν
  - Όλες οι μεταβλητές που δηλώνονται μέσα στην συνάρτηση είναι τοπικές
    - Είναι γνωστές μόνο στην συνάρτηση
  - Παράμετροι
    - επικοινωνία πληροφοριών μεταξύ συναρτήσεων (και κυρίου προγράμματος)
    - Τοπικές μεταβλητές – τοπικά αποτελέσματα
- Οφέλη
  - Μικρότερα κομμάτια κώδικα για εξειδικευμένους υπολογισμούς βοηθούν στην εκλέπτυνση της λύσης του προβλήματος και στη διαχείριση του προγράμματος
  - Επαναχρησιμοποίηση συναρτήσεων (και χωρίς να γνωρίζουμε πως έχουν υλοποιηθεί στην βιβλιοθήκη)
  - Δεν επαναλαμβάνουμε κώδικα στο πρόγραμμα μας

Δήλωση συναρτήσεων

- Δήλωση και ορισμός συνάρτησης

  Τύπος_επιστρεφόμενης_τιμής Όνομα_συνάρτησης(Λίστα παραμέτρων)
  {
  δηλώσεις και εντολές
  }

  – Όνομα_συνάρτησης: προσδιοριστή όπως οι μεταβλητές
  – Τύπος_επιστρεφόμενης_τιμής: (εξ ορισμού **`int`**)
    - **`void`** – αν δεν επιστρέφει τίποτα
  – Λίστα παραμέτρων: δηλώνουμε τύπο και όνομα παραμέτρου, χωρίζουμε με κόμμα

Δήλωση συναρτήσεων

- ….. Δήλωση και ορισμός συνάρτησης

  Τύπος_επιστρεφόμενης_τιμής Όνομα_συνάρτησης(Λίστα παραμέτρων)
  {
  δηλώσεις και εντολές
  }

  – *δηλώσεις και εντολές*: σώμα συνάρτσξης
    - Μεταβλητές
    - Εντολές
  – Έλεγχος επιστρέφει στο σημείο κλήσεις της συνάρτησης
    - Εάν δεν επιστρέφει τιμή
      – Μόλις εκτελεστεί η **return;**
      – Η όταν φτάσει στη τελευταία δεξιά «}»
    - Εάν επιστρέφει τιμή
      – Μόλις εκτελεστεί η **return** *expression*;

Πρότυπα συναρτήσεων

- Πρότυπο συνάρτησης
  - Όνομα συνάρτησης
  - Παράμετροι
  - Τύπος επιστρεφόμενης τιμής
  - Πρότυπο χρειάζεται μόνο αν η δήλωση της συνάρτησης δίνεται μετά την κλήση της στο πρόγραμμα
  - Παράδειγμα

```
int maximum( int, int, int );
```

    - παράμετροι είναι 3 **ακέραιοι**
    - Επιστρέφει ακέραιο
    - Ονόματα παραμέτρων μπορούν να παραλειφθούν αλλά μπορούν αν δοθούν για καλύτερη αναγνωσιμότητα κώδικα

```
int maximum( int x, int y, int z );
```

Παράμετροι

- Τυπικές / πραγματικές παράμετροι
  - Η παράμετροι που ορίζονται στην δήλωση των συναρτήσεων ονομάζονται τυπικές παράμετροι εν αντιθέσει με την παράμετρο που χρησιμοποιούμε στην κλήση της συνάρτησης η οποίες ονομάζονται πραγματικές παράμετροι. Οι δύο αυτοί τύποι παραμέτρων πρέπει να συμφωνούν σε τύπο, σειρά και πλήθος. Κατά την κλήση η τιμή της πραγματικής περνάει στην τυπική παράμετρο. (Υπάρχει και άλλος τρόπος να περάσουμε μία παράμετρο κατά την κλήση όπου περνάμε διεύθυνση και όχι τιμή).
  - Οι μεταβλητές που δηλώνονται μέσα στην συνάρτηση ονομάζονται τοπικές εν αντιθέσει με αυτές που ορίζονται στο κύριο πρόγραμμα και ονομάζονται καθολικές. Οι τοπικές είναι ορατές μόνο μέσα στην συνάρτηση.

Παράδειγμα

```
#include <iostream>
#include <cstdlib>
int maximum( int, int, int );   /* πρότυπο */
using namespace std;
int main(int argc, char *argv[])
{ system("chcp 1253");
  int a, b, c;
   cout<<"Δόσε 3 ακεραίους: ";
   cin>>a>>b>>c;
   cout<<"Max είναι: "<<maximum( a, b, c )<<endl; //κλήση
   system("PAUSE");
   return 0;
}
/* δήλωση συνάρτησης */
int maximum( int x, int y, int z )
{  int max = x;
   if ( y > max )  max = y;
   if ( z > max )  max = z;
   return max;
}
```

Αρχεία Header

- Αρχεία επικεφαλίδες
  - Περιέχουν πρότυπα για συναρτήσεις σε βιβλιοθήκες
  - **`<cstdlib>`** , **`<cmath>`** , κ.α.
  - **`#include <cmath>`**

Κλήση συναρτήσεων: Παράμετροι τιμής, Παράμετροι αναφοράς

- Πέρασμα παραμέτρων
- Παράμετροι τιμής
  - Η τιμής της πραγματικής παραμέτρου περνάει στην συνάρτηση
  - Αλλαγές στην συνάρτηση δεν επηρεάζουν την πραγματική παράμετρο
  - Χρησιμοποιείται όταν δεν χρειάζεται να αλλάξουμε την παράμετρο και είναι καλό γιατί αποφεύγουμε αλλαγές κατά λάθος
- Παράμετροι αναφοράς
  - Περνάμε την διεύθυνση της πραγματικής παραμέτρου
  - Τυχόν αλλαγές μέσα στην συνάρτηση επηρεάζουν την τιμή της πραγματικής έξω από την συνάρτηση
  - Χρησιμοποιείται όταν όντως θέλουμε να αλλάξουμε την πραγματική παράμετρο (είναι και ένας άλλος τρόπος επιστροφής αποτελεσμάτων). Χρειάζεται προσοχή. Υπάρχει ειδικός συντακτικός μηχανισμός για αυτές τις παραμέτρους.
- Προς το παρόν χρησιμοποιούμε και επικεντρωνόμαστε μόνο στις παραμέτρους τιμής

Τυχαίοι αριθμοί

- **`rand`** function
  - #include **`<cstdlib>`**
  - Επιστρέφει ένα «ψευδό-τυχαίο" αριθμό μεταξύ **`0`** και **`RAND_MAX`** (τουλάχιστον **`32767`**)

```
i = rand();
```

  - Ψευδό-τυχαίοι
    - Προκαθορισμένη ακολουθία από τυχαίους
    - Ίδια ακολουθία κάθε φορά που τρέχει το πρόγραμμα αλλά επηρεάζεται από την αρχική τιμή (αρχικοποίηση) η οποία μπορεί να είναι τυχαία τιμή (π.χ. ρολόι του συστήματος)
- Αλλαγή διαστήματος
  - Για [1, **`n] έχουμε  1 + ( rand() % n )`**
    - **`rand() % n`** → [**`0, n - 1]`**
    - Συν **`1`** → [**`1, n]`**
    - **`1 + ( rand() % 6)`** → **`[1, 6]`**

Τυχαίοι αριθμοί

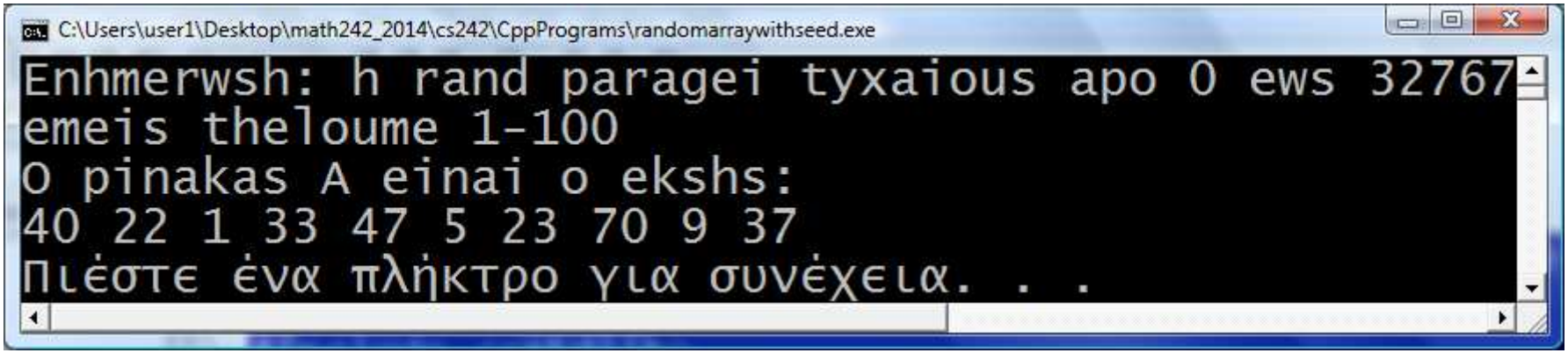
- **`srand`**
  - **`<cstdlib>`**
  - Αρχικοποιείται με ένα αρχικό τυχαίο ακέραιο αλλάζοντας έτσι την ακολουθία των τυχαίων που επιστρέφει κάθε φορά με την **`srand(`** *seed* **`);`**
  - **`srand(time(NULL)); //#include <ctime>`**
    - **`time( NULL )`**
      - Η τρέχουσα τιμή του ρολογιού σε δευτερόλεπτα
      - Τυχαιοποιεί την αρχική τιμή για την γεννήτρια τυχαίων αριθμών

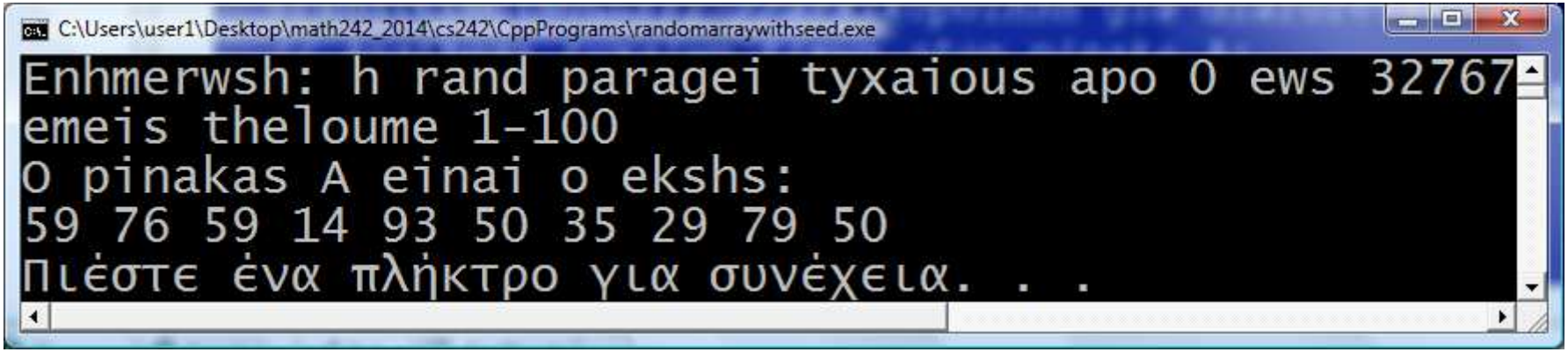
Τυχαίοι αριθμοί

```
/* random 1-dimensional array - seeded random numbers*/
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

int main(int argc, char *argv[])
{ int i, N, A[10];
  N = 10;
  srand(time(NULL)); //arxikopoihsh gia diaforetikous tyxaious ka8e fora
  // Anathesi tyxaiwn timwn ston pinaka A:
  cout<<"Enhmerwsh: h rand paragei tyxaious apo 0 ews " <<RAND_MAX<<endl;
  cout<<"emeis theloume 1-100"<<endl;
  for(i=0; i<N; i++)
    { A[i] = rand() % 100 + 1; }
    //Ektypwsi tou pinaka A:
  cout << "O pinakas A einai o ekshs:"<< endl;
  for(i=0; i<N; i++)
    { cout << A[i] << " "; }
  cout << endl;
  system("Pause");
  return 0;
}
```

Τυχαίοι αριθμοί





Συναρτήσεις οριζόμενες από τον χρήστη

- Η κύρια χρήση για μία συνάρτηση είναι να υπολογίζει και να επιστρέφει μια τιμή βασιζόμενη στις παραμέτρους που περνάει ο χρήστης.
- Όμως μια συνάρτηση μπορεί να μην έχει παραμέτρους και να μην επιστρέφει τιμή
  - Παράδειγμα, συναρτήσεις που τυπώνουν μηνύματα

Συναρτήσεις οριζόμενες από τον χρήστη

```
// aplo paradeigma synarthshs xwris parametrous
#include <iostream>
#include <cstdlib>
using namespace std;

void print_message1(void); // prototype
void print_message2(); // prototype

int main(int argc, char *argv[])
{ system("chcp 1253");

  print_message1();
  print_message2();
  system("PAUSE");
  return 0;
}

void print_message1()  // dhlwsh synarthshs
{
 cout<<"Prwto mhnyma apo synarthsh print_message1.\n";
}
void print_message2()  // dhlwsh synarthshs
{
 cout<<"Deytero mhnyma apo synarthsh print_message2.\n";
}
```

Συναρτήσεις οριζόμενες από τον χρήστη

- Η κύρια χρήση για μία συνάρτηση είναι να υπολογίζει και να επιστρέφει μια τιμή βασιζόμενη στις παραμέτρους που περνάει ο χρήστης.
  - Παράδειγμα, ο χρήστης δίνει 2 ακεραίους και η συνάρτηση επιστρέφει το γινόμενο

Γινόμενο 2 ακεραίων

```
// Ginomeno 2 akeraiwn me synarthsh
#include <iostream>
#include <cstdlib>
using namespace std;

// dhlwsh synarthshs
int mult ( int x, int y )
{
  return x * y;
}

int main(int argc, char *argv[])
{ system("chcp 1253");
  int x,y,z;
  cout<<"Dose 2 arithmous gia na pollaplasiasw: ";
  cin>> x >> y;
  z=mult (x, y); //klhsh synarthshs
  cout<<"To ginomeno einai: "<<z<<"\n";
  system("PAUSE");
  return 0;
}
```

Γινόμενο 2 ακεραίων με χρήση πρωτοτύπου

```
// Ginomeno 2 akeraiwn me synarthsh
#include <iostream>
#include <cstdlib>
using namespace std;

int mult ( int x, int y ); // prototypo

int main(int argc, char *argv[])
{ system("chcp 1253");
  int x,y,z;
  cout<<"Dose 2 arithmous gia na pollaplasiasw: ";
  cin>> x >> y;
  z=mult (x, y); //klhsh synarthshs
  cout<<"To ginomeno einai: "<<z<<"\n";
  system("PAUSE");
  return 0;
}

// dhlwsh synarthshs
int mult ( int x, int y )
{
  return x * y;
}
```

Μέγιστος 3 ακεραίων

```
// max  3 akeraiwn me synarthsh
#include <iostream>
#include <cstdlib>
using namespace std;

int maximum(int, int, int);   //prototype
// max  3 akeraiwn
int main(int argc, char *argv[])
{ int a, b, c;
  cout<<"Dose 3 akeraious: ";
  cin>>a>>b>>c;
  cout<<"Max einai: "<<maximum(a,b,c)<<endl;
  system("PAUSE");
  return 0;
}

// dhlwsh synarthshs
int maximum(int x, int y, int z)
{int max = x;
 if ( y > max )  max = y;
 if ( z > max )  max = z;
 return max;
}
```

Κλάσεις μεταβλητών

- Τοπικές μεταβλητές
  - Ορίζονται μέσα σε συναρτήσεις (ή σε blocks κώδικα όπως π.χ ανακυκλώσεις for)
  - Είναι ορατές και μπορούν να χρησιμοποιηθούν μόνο μέσα στην συνάρτηση
  - Με το τέλος της κλήσης της συνάρτησης παύουν να υπάρχουν (εκτός αν δηλωθούν σαν static)
- Καθολικές μεταβλητές
  - Ορίζονται στην αρχή του προγράμματος
  - Είναι ορατές σε όλο το πρόγραμμα αλλά μέσα σε μία συνάρτηση αν υπάρχει μια τοπική μεταβλητή με το ίδιο όνομα τότε χρησιμοποιείται η τοπική μεταβλητή

Εμβέλεια - Ορατότητα

- Γενικός κανόνας για αναζήτηση ενός προσδιοριστή (όνομα μεταβλητής ή συνάρτησης) που χρησιμοποιείται σε μία παράσταση κατά την εκτέλεση μια κλήσης συνάρτησης
  - Κοιτάμε μέσα στην συνάρτηση (στις τοπικές μεταβλητές και τις παραμέτρους)
  - Μετά κοιτάμε στις περικλείνουσες συναρτήσεις από την εσωτερικότερη προς την εξωτερικότερη
  - Τέλος κοιτάμε στις καθολικές μεταβλητές
- Θεωρούμε τις συναρτήσεις (και τα blocks) σαν κουτιά με τοιχώματα που είναι μονόδρομοι καθρέπτες. Μπορούμε να δούμε προς τα έξω αλλά από έξω δεν μπορούμε να δούμε μέσα στο κουτί.

Παράδειγμα - ψηφία ακεραίου

- Να γράψετε πρόγραμμα που διαβάζει έναν ακέραιο αριθμό x και υπολογίζει το πλήθος των ψηφίων του αριθμού x. Χρησιμοποιήστε επαναληπτικές διαιρέσεις του αριθμού x με το 10

```
int x, k;
cout << "Dose arithmo "<<endl;    cin >> x;
k = 0 ;
while(x != 0)
{    x = x / 10;
     k = k + 1;  }
cout << "Ta psifia tou "<< x << " einai " << k <<endl;
```

Παράδειγμα συνάρτηση - ψηφία ακεραίου

- Να γράψετε πρόγραμμα που διαβάζει έναν ακέραιο αριθμό x και υπολογίζει το πλήθος των ψηφίων του αριθμού x.
- int CountDigits(int n) // synarthsh gia metrhma psifiwn
- {int k=0;
- while(n != 0)
- {n = n / 10;
- k = k + 1;
- }
- return k;
- }

Παράδειγμα συνάρτηση - ψηφία ακεραίου

- Να γράψετε πρόγραμμα που διαβάζει έναν ακέραιο αριθμό x και υπολογίζει το πλήθος των ψηφίων του αριθμού x.

```
int main(int argc, char *argv[]) //kyrio programma
{ int x,k;
  cout<<"Dose akeraio x: ";
  cin>>x;
  k=CountDigits(x);
  cout << "To plithos twn psifiwn tou "<< x << " einai " << k <<endl;
  system("PAUSE");   return 0;
}
```

Παράδειγμα – x^n

- Να γράψετε πρόγραμμα που διαβάζει δύο ακεραίους x και n και υπολογίζει την ύψωση σε δύναμη, δηλαδή τον αριθμό x^n. Θα πρέπει να ελέγχετε όλες τις περιπτώσεις, αν n>0, αν n=0, αν n < 0.

```
cout << "Dose vash"<<endl;    cin >> x;
cout << "Dose ektheth"<<endl;   cin >> n;
m=n;   if(n<0)   n = -n;
p = 1;
for(i=1; i<=n; i++)
        {  p = p * x;   }
if(m<0)  p = 1 / p;        // prosoxi: float p;
cout << “Ypsosi se dinami " << p <<endl;
```

Παράδειγμα συνάρτηση - x^n

- Να γράψετε πρόγραμμα που διαβάζει δύο ακεραίους x και n και υπολογίζει την ύψωση σε δύναμη, δηλαδή τον αριθμό x^n. Θα πρέπει να ελέγχετε όλες τις περιπτώσεις, αν n>0, αν n=0, αν n < 0.

```
double xpowern(double x, int n) // synarthsh gia x^n, n>=0
{double p = 1;
 for(int i=1; i<=n; i++)
    {  p = p * x;   }
 return p;
}
```

Παράδειγμα συνάρτηση – x^n

- Να γράψετε πρόγραμμα που διαβάζει δύο ακεραίους x και n και υπολογίζει την ύψωση σε δύναμη, δηλαδή τον αριθμό x^n. Θα πρέπει να ελέγχετε όλες τις περιπτώσεις, αν n>0, αν n=0, αν n < 0.

```
int main(int argc, char *argv[]) // kyrio programma
{int x,n,m;  double p;
 cout << "Dose vash: "; cin >> x;
 cout << "Dose ektheth: "; cin >> n;
 m=n;   if(n<0) n = -n;
 p=xpowern(x,n);  if(m<0)  p = 1 / p;
 cout << "To x^n einai "<< p <<endl;
 system("PAUSE");  return 0;
}
```

Παραγοντικό!

```
// Paragontiko
#include <iostream>
#include <cstdlib>
using namespace std;

long int factorial(int n)
{ long int fact = 1;
   for(int i = 1; i <= n; ++i)  fact *= i;
   return fact;
}

int main(int argc, char *argv[])
{ int n;
  long int f;
  cout<<"Dose n <=12: ";
  cin>>n;
  f=factorial(n);
  cout<<n<<"! = "<<f<<endl;
  system("PAUSE");
  return 0;
}
```

Βαθμοί Fahrenheit

```
// fahrenheit <-> Celcius
#include <iostream>
#include <cstdlib>
#define COLD 0
#define HOT 100
using namespace std;
void hotcold(int);// anagnwrizei shmeia phkshs kai thkshs
int CtoF(int);// celcius -> farenheit
int main(int argc, char *argv[])
{ int count;
  int fahrenheit;
  int celcius;
  cout<<"Table celius - farenheit\n";
  for (count=-4; count<=12; count=count+1)
      {celcius = 10 * count;
       fahrenheit = CtoF(celcius);
       cout<<" C = "<<celcius<<" F = "<<fahrenheit;
       hotcold(celcius);
      }
  system("PAUSE");
  return 0;
}
```

Βαθμοί Fahrenheit

- …… συνέχεια
- /* synarthsh gia mhnymata gia 0, 100 bathmous */
- void hotcold(int temperature)
- { if (temperature == COLD)
- cout<<" <= Freezing point of water\n";
- else if (temperature == HOT)
- cout<<" <= Boiling point of water\n";
- else cout<<"\n";
- }
- // synarthsh gia metatroph
- int CtoF(int celcius)
- {return (32 + (celcius * 9)/5);}

Αναδρομή, Αναδρομικές συναρτήσεις

- Οι αναδρομικές συναρτήσεις
  - Καλούν τον εαυτό τους
  - Πρέπει να υπάρχει μία βάση της αναδρομής για να σταματούν οι αναδρομικές κλήσεις
  - Το πρόβλημα ορίζεται αναδρομικά και κάθε αναδρομική κλήση λύνει το ίδιο πρόβλημα αλλά με μικρότερο μέγεθος
  - Σε κάθε αναδρομική κλήση, ένα νέο αντίγραφο της συνάρτησης αρχίζει να εκτελείται
  - Έτσι κάποια στιγμή θα φτάσει στην βάση της αναδρομής (η οποία λύνεται εύκολα) και θα αρχίσει να επιστρέφει αποτελέσματα πίσω προς την αρχική κλήση η οποία και θα δώσει την ολική λύση στο πρόβλημα

Παραγοντικό! (ξανά)

- Παραγοντικό
  - **5! = 5 * 4 * 3 * 2 * 1**
  - Ισχύει ότι
    - **5! = 5 * 4!**
    - **4! = 4 * 3!** …
  - Μπορούμε να υπολογίσουμε παραγοντικά αναδρομικά
  - Βάση (**1! = 0! = 1**) και μετά
    - **2! = 2 * 1! = 2 * 1 = 2;**
    - **3! = 3 * 2! = 3 * 2 = 6;**

```
long int factorial(int n)
{if (n==0) return 1;
  else return n*factorial(n-1);}
```

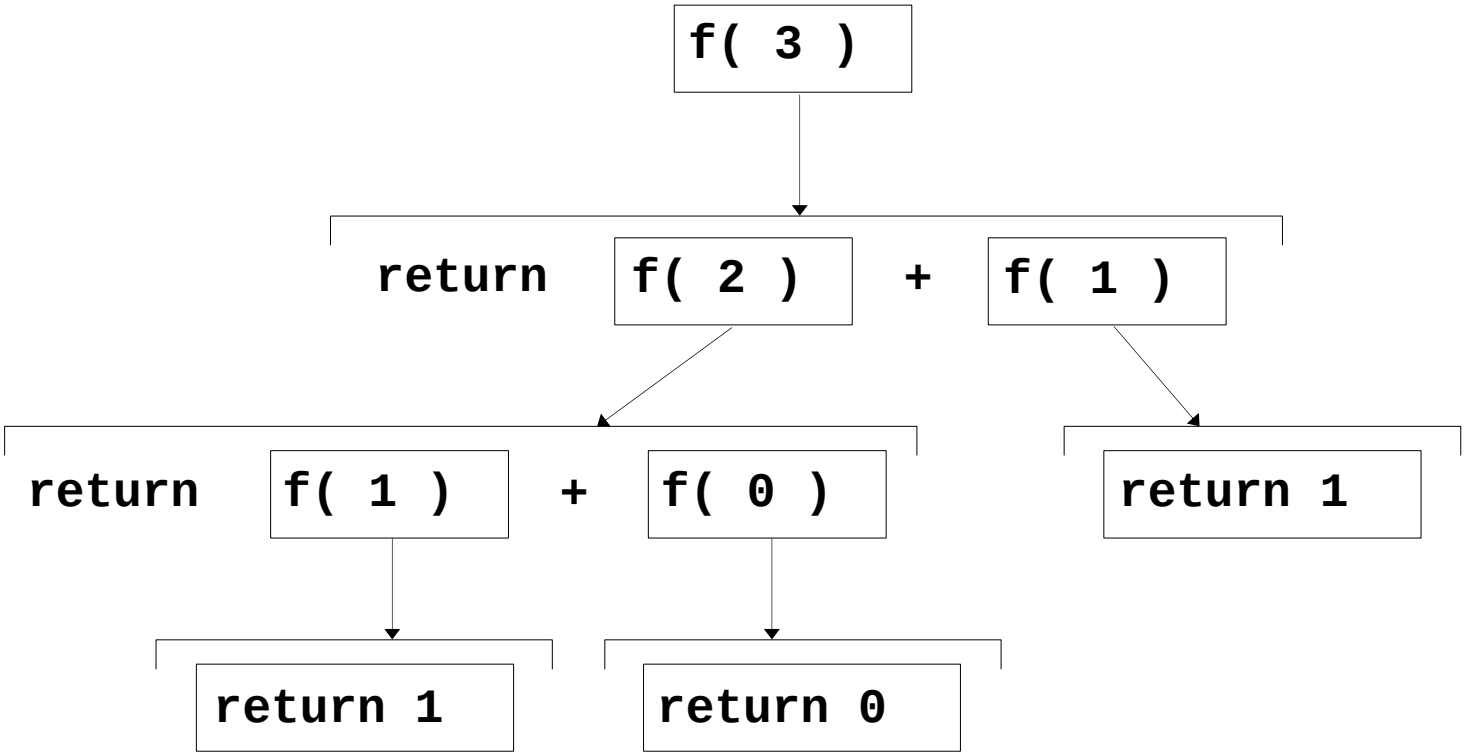
Η σειρά Fibonacci

- Fibonacci: 0, 1, 1, 2, 3, 5, 8...
  - $F_n = F_{n-1} + F_{n-2}$, $F_0=0$, $F_1=1$
  - Αναδρομική λύση:
    - **fib(n) = fib(n - 1) + fib(n – 2)**
  - Συνάρτηση **fibaonacci**

```
long fibonacci( long n )
{
  if (n == 0 || n == 1)  // βάση
     return n;
  else
     return fibonacci( n - 1) +
        fibonacci( n – 2 );
}
```

Η σειρά Fibonacci

- Αναδρομικές κλήσεις **`fibonacci`**



Η σειρά Fibonacci

```
/* Fibonacci[N] gia N<=45,
   anadromikh kai epanalhptikh synarthsh
   f(i-1), f(i), f(i+1) fib arithmoi
   0,1,1,2,3,5,8,13, ..... */
#include <iostream>
using namespace std;
int Fib(int); // prwtotype
int FibRec(int); // prwtotype anadromikh synarthsh

int main(int argc, char *argv[])
{ int n, ans;
  cout<<"Dose to deikth N tou arithmou \n";
  cout<<"pou anazhtame sthn akolou8ia Fib: ";
  cin>>n;
  ans=Fib(n);
  cout << "O "<<n<<"-os Fibonacci (epanalhptika) einai o " <<ans<<endl;
  ans=FibRec(n);
  cout << "O "<<n<<"-os Fibonacci (anadromika) einai o " <<ans<<endl;
  system("Pause");
  return 0;
}
```

Η σειρά Fibonacci

```
…. ….. συνέχεια
// epanalhptikh synarthsh
int Fib(int n)
{ int i, fn_2=0, fn_1=1, fn=1;
  i=2; // Trexwn deikths arithmou Fib sthn akolouthia
  while (i != n)
   { fn_2=fn_1;
     fn_1=fn;
     fn=fn_2+fn_1;
     i++;
   }
return fn;
}
// anadromikh synarthsh
int FibRec(int n)
{ if (n==0) return 0;
  else if (n==1) return 1;
       else return (Fib(n-2)+Fib(n-1));
}
```

Αναδρομή/Επανάληψη

- Επανάληψη υπολογισμών
  - Ανακυκλώσεις: for, while, .. (πιο γρήγορο)
  - Αναδρομή: χρήση αναδρομικών κλήσεων (πιο αργό)
- Τέλος υπολογισμών
  - Ανακυκλώσεις : συνθήκες
  - Αναδρομή : βάση αναδρομής
- Και οι δυο τρόποι μπορεί να έχουν αέναες ανακυκλώσεις αν δεν προσέξουμε
- Υπέρ/Κατά
  - Ανακυκλώσεις: for, while, .. (πιο γρήγορο αλλά μπορεί να είναι πιο δυσνόητο)
  - Αναδρομή: χρήση αναδρομικών κλήσεων (πιο αργό αλλά πιο εκφραστικό)