Γενικά

- Όλες οι προηγούμενες παραλλαγές στις εκφωνήσεις απαιτούν και διαφορετικό τρόπο επίλυσης
  - Πρόγραμμα
  - Πρόγραμμα με συναρτήσεις
  - Πρόγραμμα με συναρτήσεις και έλεγχο δεδομένων
  - Πρόγραμμα με επιμέρους συναρτήσεις (προσοχή στις &παραμέτρους)
  - Πρόγραμμα με επιμέρους συναρτήσεις και επαναληπτικό υπολογισμό

- Θα πρέπει να τους καταλαβαίνουμε από την εκφώνηση ποιο ολοκληρωμένο πρόγραμμα ζητάμε
- Στα υπόλοιπα παραδείγματα μόνο κάποια κατηγορία ζητάμε και επιλύουμε
  - Θα πρέπει ωστόσο να μπορούμε να διαχειριστούμε και τις υπόλοιπες κατηγορίες

Τετράγωνο αριθμού

- Το τετράγωνο ενός ακεραίου αριθμού n μπορεί να υπολογιστεί προσθέτοντας όλους τους ακέραιους από το 1 έως το n και επιστρέφοντας πάλι πίσω στο 1,
- π.χ. $4^2 = 1 + 2 + 3 + 4 + 3 + 2 + 1 = 16$

- Να γράψετε ένα πρόγραμμα με συναρτήσεις που θα εμφανίζει το τετράγωνο οποιουδήποτε ακέραιου n χρησιμοποιώντας τον παραπάνω τρόπο
- Ο αλγόριθμος θα δέχεται ως είσοδο τον αριθμό n.
- Κατά την εισαγωγή θα ελέγχεται η συνθήκη $n > 0$.

```
#include <iostream>
int square(int n);
int main( )
{
    int n, sqn;
    do
    {
       cout << "Enter n:";
       cin >> n;
    }
    while( n < 0 );
    cout << "Square: " << square(n);
}
int square(int n)
{
    int res;
    res = 0;
    for(int i = 1; i <= n; i++)
        res = res + i;
    for(int i = n-1; i >= 1; i--)
        res = res + i;
   return res;
}
```

Μέγιστος Κοινός Διαιρέτης

- Να γραφεί πρόγραμμα που να υπολογίζει τον *μέγιστο κοινό διαιρέτη* μεταξύ δύο ακεραίων.
- Σύμφωνα με τον αλγόριθμο του Ευκλείδη:
    1. Αν οι δύο αριθμοί είναι ίσοι, ο μέγιστος κοινός διαιρέτης ισούται μ' αυτούς.
    2. Αν δεν είναι ίσοι, αντικαθιστούμε τον μεγαλύτερο με τη διαφορά τους και επαναλαμβάνουμε τη σύγκριση

36 84 Δεν είναι ίσοι. Το 84 αντικαθίσταται με τη διαφορά 84-36=48

36 48 Δεν είναι ίσοι. Το 48 αντικαθίσταται με τη διαφορά 48-36=12

36 12 Δεν είναι ίσοι. Το 36 αντικαθίσταται με τη διαφορά 36-12=24

24 12 Δεν είναι ίσοι. Το 24 αντικαθίσταται με τη διαφορά 24-12=12

12 12 Είναι ίσοι. Ο μέγιστος κοινός διαιρέτης των 36 και 84 είναι το 12.

```
#include <iostream>
int mkd(int a, int b);
int main( )
{
    int a, b, mkd_ab;
    cout << "Enter a b:";
    cin >> a >> b;
    mkd_ab = mkd(a, b);
    cout << "MKD: " << mkd_ab;
}

int mkd(int a, int b)
{
   while( a != b)
   {
        if( a > b )
             a = a-b;
        else
             b = b-a;
   }
   return (a); // ή return (b);
}
```

Μέγιστος Κοινός Διαιρέτης

- Να γραφεί πρόγραμμα που να υπολογίζει τον *μέγιστο κοινό διαιρέτη* μεταξύ δύο ακεραίων.
- Σύμφωνα με τον αλγόριθμο του Ευκλείδη:
    1. Αν οι δύο αριθμοί είναι ίσοι, ο μέγιστος κοινός διαιρέτης ισούται μ' αυτούς.
    2. Αν δεν είναι ίσοι, αντικαθιστούμε τον μεγαλύτερο με τη διαφορά τους και επαναλαμβάνουμε τη σύγκριση

36 84 Δεν είναι ίσοι. Το 84 αντικαθίσταται με τη διαφορά 84-36=48
36 48 Δεν είναι ίσοι. Το 48 αντικαθίσταται με τη διαφορά 48-36=12
36 12 Δεν είναι ίσοι. Το 36 αντικαθίσταται με τη διαφορά 36-12=24
24 12 Δεν είναι ίσοι. Το 24 αντικαθίσταται με τη διαφορά 24-12=12
12 12 Είναι ίσοι. Ο μέγιστος κοινός διαιρέτης των 36 και 84 είναι το 12.

- **Θέλουμε να εκτυπώνουμε και το πλήθος των επαναλήψεων**

```
#include <iostream>
int mkd(int a, int b);
int main( )
{
    int a, b, mkd_ab;
    cout << "Enter a b:";
    cin >> a >> b;
    mkd_ab = mkd(a, b);
    cout << "MKD: " << mkd_ab;
}

int mkd(int a, int b)
{
   while( a != b)
   {
        if( a > b )
             a = a-b;
        else
             b = b-a;
   }
   return (a); // ή return (b);
}
```

```
#include <iostream>
int mkd(int a, int b, int &k);
int main( )
{
    int a, b, mkd_ab, n = 0;
    cout << "Enter a b:";
    cin >> a >> b;
    mkd_ab = mkd(a, b, n);
    cout << "MKD: " << mkd_ab << "n=" << n;
}

int mkd(int a, int b, int &k)
{
   while( a != b)
   {
        if( a > b )
             a = a-b;
        else
              b = b-a;
         k = k + 1;
   }
   return (a); // ή return (b);
}
```

Μέγιστος Κοινός Διαιρέτης (αναδρομικά)

- Να γραφεί πρόγραμμα με αναδρομική συνάρτηση που να υπολογίζει τον *μέγιστο κοινό διαιρέτη* μεταξύ δύο ακεραίων.

- Σκεφτείτε το υπόλοιπο της διαίρεσης x % y :

- Έστω ότι x > y
  - Εάν y = 0 τότε ΜΚΔ(x,y)=x
  - Αλλιώς, ΜΚΔ(x,y)=ΜΚΔ(y, x%y)

```
#include <iostream>
int mkd(int a, int b);
int main( )
{
    int a, b, mkd_ab;
    cout << "Enter a b:";
    cin >> a >> b;

    if( a > b )
       mkd_ab = mkd(a, b);
    else
       mkd_ab = mkd(b, a);

    cout << "MKD: " << mkd_ab;
}

int mkd(int a, int b)
{
   if( b == 0 )
      return a;
   else
      return ( mkd( b, (a % b) ) );
}
```

Υπολογισμός δύναμης

- Να γραφτεί πρόγραμμα που διαβάζει δύο ακεραίους x και y και στη συνέχεια υπολογίζει την έκφραση x^y.

[χωρίς τη χρήση
της έτοιμης
συνάρτησης pow()]

- Παρατήρηση:

  $x^y = x\ x\ x\ \ldots\ x\ x$

Υπολογισμός δύναμης

- Να γραφτεί πρόγραμμα που διαβάζει δύο ακεραίους x και y και στη συνέχεια υπολογίζει την έκφραση x^y.

[χωρίς τη χρήση της έτοιμης συνάρτησης pow()]

- Παρατήρηση:

  $x^y = x \; x \; x \; \ldots \; x \; x$

```
#include <iostream>


int main( )
{
    int x, y, i, result;

    cout << "Enter x y:";
    cin >> x >> y;

    result = 1;
    for( i = 1; i <= y; i++)
       result = result * x;

    cout << result;
}
```

Υπολογισμός δύναμης

- Τι υπολογίζεται με τις ακόλουθες εντολές:

```
for(i = 0; i < y; i++)

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

for(i = y; i >= 0; i--)

for(i = y; i > 0; i--)
```

```
#include <iostream>


int main( )
{
    int x, y, i, result;

    cout << "Enter x y:";
    cin >> x >> y;

    result = 1;
    for( i = 1; i <= y; i++)
       result = result * x;

    cout << result;
}
```

Υπολογισμός δύναμης

- Βεβαιωθείτε ότι το πρόγραμμα εκτελείται σωστά για

  θετικές ή αρνητικές τιμές για την βάση και για

  θετικές ή αρνητικές τιμές για τον εκθέτη.
  - x= 4, y = -2 ?

```
#include <iostream>


int main( )
{
    int x, y, i, result;

    cout << "Enter x y:";
    cin >> x >> y;

    result = 1;
    for( i = 1; i <= y; i++)
       result = result * x;

    cout << result;
}
```

Υπολογισμός δύναμης

- Βεβαιωθείτε ότι το πρόγραμμα εκτελείται σωστά για

  θετικές ή αρνητικές τιμές για την βάση και για

  θετικές ή αρνητικές τιμές για τον εκθέτη.
  - x= 4, y = -2 ?
- $x^{-y} = 1 / x^{y}$

1. Υπολογίζουμε το $x^{|y|}$ και
2. εξετάζουμε αν το y είναι αρνητικό για να υπολογίσουμε το $1 / x^{y}$.

```
#include <iostream>


int main( )
{
    int x, y, i, result;

    cout << "Enter x y:";
    cin >> x >> y;

    result = 1;
    for( i = 1; i <= y; i++)
       result = result * x;

    cout << result;
}
```

Υπολογισμός δύναμης

- Βεβαιωθείτε ότι το πρόγραμμα εκτελείται σωστά για

  θετικές ή αρνητικές τιμές για την βάση και για

  θετικές ή αρνητικές τιμές για τον εκθέτη.
  - x= 4, y = -2 ?
- $x^{-y} = 1 / x^{y}$

1. Υπολογίζουμε το $x^{|y|}$ και
2. εξετάζουμε αν το y είναι αρνητικό για να υπολογίσουμε το $1 / x^{y}$.

```
#include <iostream>
#include <cstdlib>

int main( )
{
    int x, y, i;
    double result;

    cout << "Enter x y:";
    cin >> x >> y;

    result = 1.0;
    for( i = 1; i <= abs(y); i++)
       result = result * x;
    if( y < 0)
        result = 1.0/result;
    cout << result;
}
```

Υπολογισμός δύναμης με αναδρομή

- Να γραφεί αναδρομική συνάρτηση που δέχεται 2 ακεραίους a, b και υπολογίζει (επιστρέφει) το a^b.
- $a^b = a * (a^{b-1})$

Υπολογισμός δύναμης με αναδρομή

- Να γραφεί αναδρομική συνάρτηση που δέχεται 2 ακεραίους a, b και υπολογίζει (επιστρέφει) το a^b.
- $a^b = a * (a^{b-1})$

- Αν b=10 πόσες κλήσεις (επαναλήψεις) θα κάνει;

```
int power( int a, int b )
{
    if(b == 0)
        return 1;
    else
        return (a * power(a,b-1) );
}
```

Υπολογισμός δύναμης με αναδρομή

- Να γραφεί αναδρομική συνάρτηση που δέχεται 2 ακεραίους a, b και υπολογίζει (επιστρέφει) το a^b.
- $a^b = a * (a^{b-1})$

```
int power( int a, int b )
{
    if(b == 0)
        return 1;
    else
        return (a * power(a,b-1) );
}
```

- Αν b=10 πόσες κλήσεις (επαναλήψεις) θα κάνει;

- Πιο γρήγορα:

$$a^b = (a^{b/2}) * (a^{b/2})$$

Υπολογισμός δύναμης με αναδρομή

- Να γραφεί αναδρομική συνάρτηση που δέχεται 2 ακεραίους a, b και υπολογίζει (επιστρέφει) το a^b.
- $a^b = a * (a^{b-1})$

```
int power( int a, int b )
{
    if(b == 0)
        return 1;
    if( b % 2 == 0 )
       return ( power(a,b/2) *
                    power(a,b/2) );
    else
        return ( power(a,b/2) *
                    power(a,b/2) * a );
}
```

- Αν b=10 πόσες κλήσεις (επαναλήψεις) θα κάνει;

- Πιο γρήγορα:

$$a^b = (a^{b/2}) * (a^{b/2})$$

$a^b = (a^{b/2}) * (a^{b/2})$, για b άρτιο

$a^b = (a^{b/2}) * (a^{b/2}) * a$, για b περιττό

#κλήσεις δεν μειώθηκαν

Υπολογισμός δύναμης με αναδρομή

- Να γραφεί αναδρομική συνάρτηση που δέχεται 2 ακεραίους a, b και υπολογίζει (επιστρέφει) το a^b.
- $a^b = a * (a^{b-1})$

```
int power( int a, int b )
{
    int half;
    if(b == 0)
        return 1;
    half = power(a, b/2) ;
    if( b % 2 == 0 )
       return ( half * half );
    else
        return (half * half * a );
}
```

- Αν b=10 πόσες κλήσεις (επαναλήψεις) θα κάνει;
- Πιο γρήγορα:

$$a^b = (a^{b/2}) * (a^{b/2})$$

$a^b = (a^{b/2}) * (a^{b/2})$, για b άρτιο

$a^b = (a^{b/2}) * (a^{b/2}) * a$, για b περιττό

Υπολογισμός αριθμητικής συνάρτησης

- Να γραφεί μια συνάρτηση που δέχεται δύο ακέραιες τιμές x και n (=άρτιος αριθμός), και υπολογίζει την τιμή της ακόλουθης αριθμητικής συνάρτησης:

$$1+1x^2+3x^4+5x^6+\cdots+(n-1)x^n$$

- main(): Καλέστε από την main() την συνάρτηση που φτιάξατε αφού πρώτα διαβάσετε τους αριθμούς. Θα πρέπει κατά την είσοδο να ελέγχετε (επαναληπτικά) για επιτρεπτές τιμές.

```
#include <iostream>
double fff(int x, int n);
int main( )
{
    int x, n;
    do
    {
       cout << "Enter x, n:";
       cin >> x >> n;
    } while ( n % 2 == 1);

    cout << "F(x,n): " << fff(x,n);
}

double fff(int x, int n)
{
   double sum = 1.0, xd = 1.0;
   for( int i = 2; i <= n; i = i + 2)
   {
      xd = xd * (x * x);
      sum = sum + (i-1) * xd;
   }
   return sum;
}
```

Φίλιοι αριθμοί

- Δύο θετικοί ακέραιοι αριθμοί είναι «φίλιοι» αν ο καθένας ισούται με το άθροισμα όσων διαιρούν τον άλλον (λαμβάνονται υπόψη μόνον οι γνήσιοι διαιρέτες). Οι πιο διάσημοι «φίλιοι» αριθμοί είναι οι αριθμοί 220 και 284 (αποδίδονται στον Πυθαγόρα).
  - Διαιρέτες του 220 : 1, 2, 4, 5, 10, 11, 20, 22, 44, 55, 110
    (άθροισμα=284)
  - Διαιρέτες του 284 : 1, 2, 4, 71, 142
    (άθροισμα=220)

- Να γραφεί μια συνάρτηση που δέχεται δύο ακεραίους και θα επιστρέφει true ή false ανάλογα αν είναι φίλιοι αριθμοί.

```
bool filioi(int a, int b)
{
    int i, sum1=0, sum2=0;

    for(i = 1; i < a; i++)
    {
        if( a%i == 0 )
            sum1 = sum1 + i;
    }

    for(i = 1; i < b; i++)
    {
        if( b%i == 0 )
            sum2 = sum2 + i;
    }

    if( sum1 == b && sum2 == a)
        return true;
    else
        return false;
}
```

```
#include <iostream>
bool filioi(int a, int b);
int main( )
{
   int a, b;

   cout << "Enter a, b:";
   cin >> a >> b;

   if( filioi(a, b) )
      cout << "Filioi!";
   else
      cout << "Not Filioi!";
}
```

Συναρτήσεις (σύνοψη)

- Όταν πρόκειται να γράψουμε μια συν/ση:
  1. Καθορίζουμε τον επιστρεφόμενο τύπο
     - Πχ. `int`
  2. Δίνουμε όνομα στην συν/ση
     - Πχ. `square`
  3. Δηλώνουμε τα ορίσματα (τι θα πρέπει να δέχεται η συν/ση ως είσοδο)
     - Πχ. `(int y)`
  4. Γράφουμε την δήλωση της συν/σης
     - Πχ. `int square(int y);`
  5. Υλοποιούμε την μέθοδο έχοντας υπόψιν την εντολή `return`
     - Πχ.

```
int square(int y)
{
        int result;
        result = y*y;
        return result;
}
```

Καλή Μελέτη

- **Βιβλιογραφία**

[1] W. Savitch, Πλήρης C++, Εκδόσεις Τζιόλα, 2011

[2] H. Deitel and P. Deitel, C++ Προγραμματισμός 6η Εκδοση, Εκδόσεις Μ. Γκιούρδας, 2013

Ύλη βιβλιογραφίας
[1]: Κεφ. 1 – 4, 13.1 - 13.2,
Παραρτήματα 1, 2, 3, 4
[2]: Κεφ. 1, 2, 4, 6
Παραρτήματα Α, Β, Γ, Δ, ΣΤ

- 1ο QUIZ (όλα τα τμήματα): **Δευτέρα 25 Νοεμβρίου**

Ανοικτά Ακαδημαϊκά Μαθήματα

Πανεπιστήμιο Ιωαννίνων

Τέλος Ενότητας

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Ιωαννίνων**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Σημειώματα

Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης