

Εξετάσεις 2024-25:

Επανάληψη C++

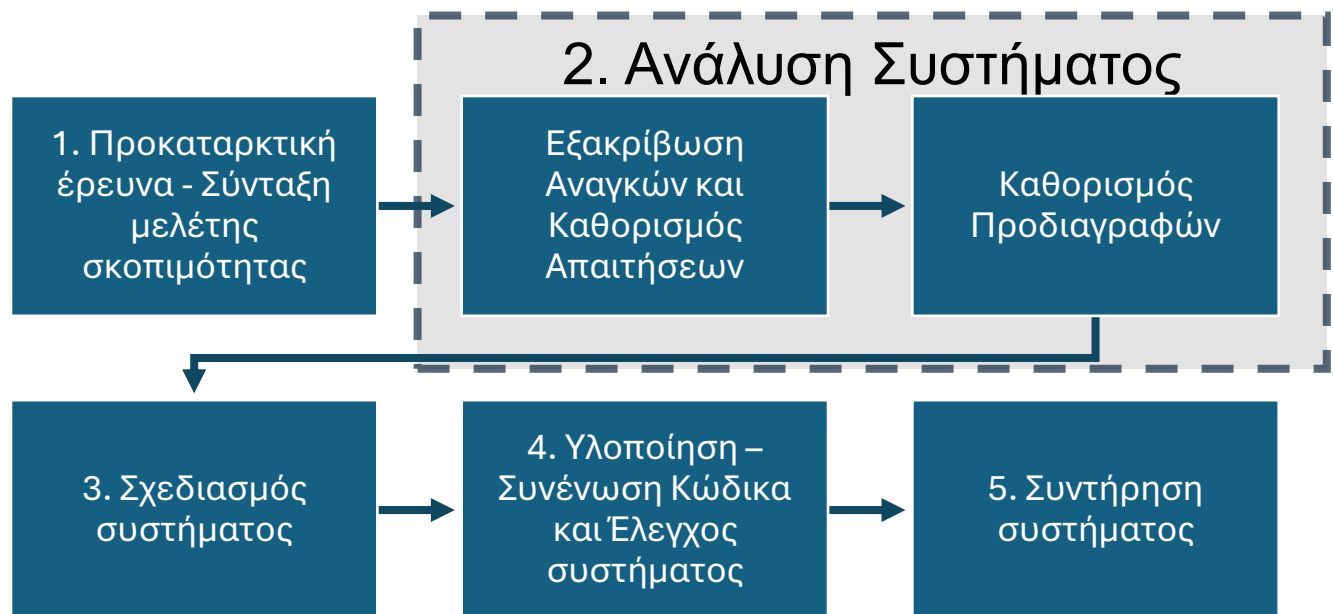
Περιεχόμενο:

Κύκλος Ανάπτυξης Προγράμματος	3
Λογικά Διαγράμματα:	4
Προκαταρκτική Εκτέλεση:	5
Παράδειγμα μετατροπής κώδικα σε προκαταρκτική εκτέλεση:	5
Βασικές λειτουργίες και βιβλιοθήκες C++:	6
Βασικοί τύποι μεταβλητών:	6
Σχόλια:	6
Προετοιμασία int main():	7
cin/cout:	7
Αριθμητικοί τελεστές:	8
Βιβλιοθήκη iomanip:	9
Βιβλιοθήκη cmath:	10
Δομή διακλάδωσης:	10
Συγκριτικοί τελεστές:	10
Λογικοί τελεστές:	10
Προτεραιότητα λογικών τελεστών:	11
if, else if, else:	12
switch:	12
Δομή Επανάληψης:	13
while:	13
do while:	13
for:	13





Πίνακες:.....	14
Ορισμός πίνακα:	14
Επεξεργασία στοιχείων του πίνακα:.....	15
Συναρτήσεις:.....	16
Αρχεία (βιβλιοθήκη fstream):	17
Εισαγωγή Βιβλιοθήκης:	17
Διαβάζοντας ένα αρχείο:	17
Επεξεργασία αρχείων:	18

Κύκλος Ανάπτυξης Προγράμματος

1. Προκαταρκτική έρευνα - Σύνταξη μελέτης σκοπιμότητας
2. Ανάλυση Συστήματος
 - Εξακρίβωση Αναγκών και Καθορισμός Απαιτήσεων
 - Καθορισμός Προδιαγραφών
3. Σχεδιασμός συστήματος
4. Υλοποίηση – Συνένωση Κώδικα και Έλεγχος συστήματος
5. Συντήρηση συστήματος



Λογικά Διαγράμματα:

Σύμβολο	Επεξήγηση
	Αρχή ή τέλος προγράμματος
	Επεξεργασία - ορισμός μεταβλητών, μαθηματικές πράξεις
	Είσοδος Δεδομένων (cin) - Έξοδος Αποτελεσμάτων (cout)
	Απόφαση - δομές διακλάδωσης (if, else, else if, switch) και δομές επανάληψης (for, while, do)

Προκαταρκτική Εκτέλεση:

Η πρώτη στήλη είναι για τις μεταβλητές, η δεύτερη για τις αποφάσεις και η τρίτη για την παρουσίαση (η οποία περιέχει οτιδήποτε κάνουμε `cout`).

Οι αποφάσεις περιλαμβάνουν την δομή διακλάδωσης (`if`, `else if`, `else`, `switch`) αλλά και την δομή επανάληψης (`for`, `while`, `do`).

Παράδειγμα μετατροπής κώδικα σε προκαταρκτική εκτέλεση:

```
#include <iostream>
using namespace std;

int main()
{
    int a=3, b=4;
    if(a>4){
        cout<<"a μεγαλύτερο του 4.";
    }
    else{
        cout<<"a μικρότερο του 4.";
    }

    if(b==4){
        cout<<"b ίσο με 4.";
    }

    return 0;
}
```

Μεταβλητές		Αποφάσεις				Παρουσίαση
a	b	a>b	True/False	b=4	True/False	a μικρότερο του 4.
3	4	3>4	FALSE	4=4	TRUE	b ίσο με 4.

Βασικές λειτουργίες και βιβλιοθήκες C++:

Βασικοί τύποι μεταβλητών:

```
// Πραγματικοί αριθμοί:

// Integer → Ακέραιος Αριθμός
// Αν διαιρεθεί και υπάρξει υπόλοιπο, το δεκαδικό μέρος θα κοπεί.
int num = 5;

// Float → Δεκαδικός Αριθμός
float decimal = 5.5;

// Boolean → Τιμή αληθής ή ψευδής (true / false):
bool isTrue = true;
/*
    Στην C++, οι μεταβλητές bool ορίζονται και με 0 ή 1
    (0=ψευδής, 1=αληθής)
*/
isTrue=0;

// char → χαρακτήρας
char symbol = 'a';

// string → Συμβολοσειρά
string message = "Hello World!";

// Σταθερές:
/* Ορίζονται συνήθως στην αρχή του προγράμματος. Αφού οριστούν, δεν
μπορούν να αλλάξουν. */

// const: Μπαίνει πριν από τον τύπο της μεταβλητής.
const int pi = 3.14;

// define: Δεν απαιτεί τύπο μεταβλητή, μόνο όνομα και τιμή.
#define isMyComputerOn true;
```

Σχόλια:

Τα σχόλια διευκολύνουν στην εξήγηση της λειτουργίας του κώδικα που γράφουμε. Με την χρήση των σχολίων διευκολύνουμε τους εαυτούς μας και όλους όσους διαβάζουν τον κώδικά μας.

```
// Αυτό είναι ένα σχόλιο.
/* Αυτό είναι επίσης ένα σχόλιο */
```

Προετοιμασία int main():

Ένα απλό πρόγραμμα C++ ξεκινά με τα παρακάτω στοιχεία:

```
#include <iostream>
using namespace std;

int main()
{
    // Εδώ θα γράψουμε τον κώδικά μας.

    return 0;
}
```

Οι εντολές `#include <iostream>` και `using namespace std;` μας δίνουν την ικανότητα να χρησιμοποιήσουμε τις εντολές `cin` & `cout`, γι' αυτό πρέπει να τις προσθέτουμε σε κάθε πρόγραμμα.

Η συνάρτηση `int main()` εκτελείται μόλις τρέξουμε το πρόγραμμά μας. Χωρίς αυτήν, το πρόγραμμά μας δεν θα κάνει τίποτα.

cin/cout:

Είναι οι βασικότερες εντολές σε ένα απλό πρόγραμμα C++.

```
int num;

/* Είσοδος πληροφοριών: Ζητάμε από τον χρήστη να δώσει έναν αριθμό,
ο οποίος θα καταχωρηθεί στην μεταβλητή num. */
cin>>num;

/* Έξοδος Πληροφοριών: Τυπώνουμε τον αριθμό μαζί με ένα μήνυμα. */
cout<<"Ο αριθμός είναι: "<<num;
```

Αριθμητικοί τελεστές:

Σύμβολο	Λειτουργία	Παράδειγμα
+	Προσθέτει	<code>int a=5; int b=a+5; // b=10</code>
-	Αφαιρεί	<code>int a=5; int b=a-5; // b=0</code>
*	Πολλαπλασιάζει	<code>int a=5; int b=a*3; // b=15</code>
/	Διαιρεί	<code>int a=6; int b=a/2; // b=3</code>
%	Βρίσκει το υπόλοιπο	<code>int a=5; int b=a%2; // b=1</code>
++	Προσθέτει 1	<code>int a=5; a++; // a=6</code>
--	Αφαιρεί 1	<code>int a=5; a--; // a=4</code>

Σύμβολο	Λειτουργία	Παράδειγμα
+=	Προσθέτει σε μία μεταβλητή	<code>int a=5; a+=5; // 5→10</code>
-=	Αφαιρεί από μία μεταβλητή	<code>int a=5; a-=5; // 5→0</code>
=	Πολλαπλασιάζει μία μεταβλητή	<code>int a=5; a=3; // 5→15</code>
/=	Διαιρεί μία μεταβλητή	<code>int a=6; a/=2; // 6→3</code>
%=	Μετατρέπει μία μεταβλητή στο υπόλοιπό της	<code>int a=5; int b=a%2; // 5→1</code>

Βιβλιοθήκη `iomanip`:

Η `iomanip` είναι μια βιβλιοθήκη που μας δίνει μερικές επιπρόσθετες λειτουργίες.

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
    float pi=3.14159;

    // Αντί για 3.14159, θα τυπώσει μόνο x δεκαδικά ψηφία.
    cout<<fixed<<setprecision(2)<<pi; // Έξοδος: "3.14"

    /*
       Η setw ελέγχει το μέγεθος της ακόλουθης συμβολοσειράς
       (string). Αν το μέγεθος της setw είναι μεγαλύτερο από
       την συμβολοσειρά, θα προστεθούν κενά για να συμπληρωθεί
       η συμβολοσειρά.
    */
    cout<<setw(4)<<"***"; // Έξοδος: "  ***"
}
```

Βιβλιοθήκη cmath:

Η cmath, μας δίνει την δυνατότητα να εκτελέσουμε πιο πολύπλοκες μαθηματικές πράξεις. Εισάγεται με την εντολή: `#include <cmath>`

Σύμβολο	Λειτουργία	Παράδειγμα
sqrt(x)	Βρίσκει την τετραγωνική ρίζα του αριθμού x.	<pre>int a=4; a=sqrt(a); // 4→2</pre>
cbrt(x)	Βρίσκει την κυβική ρίζα του αριθμού x.	<pre>int a=8; a=cbrt(a); // 8→2</pre>
pow(x, y)	Υψώνει έναν αριθμό x στην δύναμη y.	<pre>int a=5; int b=pow(a,2); // 5→25</pre>
abs(x)	Βρίσκει την απόλυτη τιμή του αριθμού x.	<pre>int a=-5; a=abs(a); // -5→5</pre>
trunc(x)	Κόβει το δεκαδικό μέρος από έναν αριθμό x.	<pre>float a=5.5365; a=trunc(a); // 5.5365→5</pre>
round(x)	Στρογγυλοποιεί έναν αριθμό x προς τον κοντινότερο ακέραιο.	<pre>float a=5.5; float b=-5.5; a=round(a); // 5.5→6 b=round(b); // -5.5→-6</pre>
floor(x)	Στρογγυλοποιεί έναν αριθμό x προς τον προηγούμενο ακέραιο. Αν ο αριθμός είναι αρνητικός θα πάει στον επόμενο αρνητικό.	<pre>float a=5.8; float b=-5.8; a=floor(a); // 5.8→5 b=floor(b); // -5.8→-6</pre>

Δομή διακλάδωσης:

Συγκριτικοί τελεστές:

Σύμβολο	Όνομα
==	Ίσο
!=	Άνισο
>	Μεγαλύτερο
<	Μικρότερο
>=	Μεγαλύτερο ή ίσο
<=	Μικρότερο ή ίσο

Λογικοί τελεστές:

Σύμβολο	Όνομα
&&	AND
	OR
!	NOT

Προτεραιότητα λογικών τελεστών:

Προτεραιότητα έχει πρώτα η &&, και μετά η ||.

Παράδειγμα 1:

```
bool a = true;
bool b = false;
bool c = false;
```

```
bool abc = a || b && c;
```

Η abc θα ισούται με: a ή (b και c)

Παράδειγμα 2:

```
bool a = true;
bool b = false;
bool c = false;
bool d = false;
```

```
bool abcd = a && b || c || d;
```

Η abcd θα ισούται με (a και b) ή c ή d

Το '!' μπορεί να αντιστρέψει κάθε σύγκριση εάν βάλουμε παρένθεση, δηλαδή:

```
int number;
cin>>number;
if(!(number>2))
    cout<<"Όχι μεγαλύτερος του 2!";
```

if, else if, else:

```
int number;
cin>>number;

if(number==1){
    cout<<"Ο αριθμός ισούται με 1.";
}
else if(number > 1){
    cout<<"Ο αριθμός είναι μεγαλύτερος του 1.";
}
else{
    /*
        Εφόσον ο αριθμός δεν είναι ούτε μεγαλύτερος,
        ούτε ίσος με 1, θα είναι μικρότερος του 1.
    */
    cout<<"Ο αριθμός είναι μικρότερος του 1.";
}
```

switch:

Η switch είναι χρήσιμη εάν ελέγχεις για πολλές τιμές. Φαίνονται πιο συγυρισμένα έτσι. Στην παρένθεση τις switch βάζεις την μεταβλητή που θες να ελέγξεις.

Έπειτα, μέσα στις αγκύλες τις switch, βάζεις τις περιπτώσεις (case), οι οποίες τελειώνουν με break, για να βγούμε από την switch.

```
int number;
cin>>number;

switch (number){
    case 1: // Αν ο αριθμός ισούται με 1
        cout<<"1";
        break;
    case 2: // Αν ο αριθμός ισούται με 2
        cout<<"2";
        break;
    // Αν ο αριθμός δεν ισούται με τα προηγούμενα.
    default: // Η τελευταία συνθήκη δεν χρειάζεται break.
        cout<<"?";
}
```

Δομή Επανάληψης:

while:

Η δομή επανάληψης while εκτελείται όσο η συνθήκη που ορίζουμε ισχύει.

Στην παρακάτω περίπτωση, όσο η μεταβλητής num είναι μικρότερη ή ίση του 5, θα τυπώνουμε την τιμή της και θα προσθέτουμε 1.

```
int num=0;

//  συνθήκη
while(num<=5){
    cout<<num<<endl;
    num++;
}
```

do while:

Η δομή επανάληψης do while είναι παρόμοια με την while, όμως εκτελείται τουλάχιστον μία φορά, ασχέτως αν η συνθήκη που ορίζουμε ισχύει.

```
int num=0;
do {
    cout<<num<<endl;
    num++;
}
//  συνθήκη
while(num<=5);
```

for:

Η for δέχεται τρεις ιδιότητες. Στην πρώτη, ορίζουμε την μεταβλητή. Στην δεύτερη την συνθήκη, και στην τρίτη μια πράξη με την μεταβλητή.

Στην παρακάτω περίπτωση, δημιουργείται η μεταβλητής i, η οποία ξεκινάει από το 0. Όσο η μεταβλητής είναι μικρότερη του 10, θα τυπώνεται το μήνυμα «Η δομή επανάληψης εκτελείται» και θα προσθέτουμε 1 στην i.

```
//  μεταβλητής  συνθήκη  πράξη
for( int i=0;    i<10;    i++ ){
    cout<<"Η δομή επανάληψης εκτελείται.";
}
```

Πίνακες:

Ορισμός πίνακα:

Οι πίνακες περιέχουν έναν αριθμό μεταβλητών του ίδιου τύπου. Ουσιαστικά είναι σαν λίστες με δεδομένα. Ο κάθε αριθμός, σύμβολο, κτλ. που περιέχει ο πίνακας ονομάζεται στοιχείο.

Ο πίνακας ορίζεται ως εξής:

τύπος μεταβλητή, όνομα μεταβλητή, [μέγεθος πίνακα], = { στοιχεία }

```
int array1[5] = {2,4,8,16,32};

/* Τα στοιχεία δεν χρειάζεται να οριστούν, όμως οι αριθμοί θα πάρουν
περίεργες τιμές. */
int array2[5];

// Πίνακας δεκαδικών
float floats[3] {5.2, 5.6, 3.5};

// Πίνακας χαρακτήρων
// Η συμβολοσειρά είναι ουσιαστικά πίνακας χαρακτήρων
char characters[3] = {'A', 'b', '#'};

// Πίνακας συμβολοσειρών
string strings[3] = {"Dave", "John", "Max"};

// Πίνακας boolean
bool booleans[3] = {true, false, false};
```

Εάν ορίσουμε απευθείας τα στοιχεία του πίνακα, το μέγεθός του ορίζεται αυτόματα:

```
// Ορίσαμε 5 στοιχεία, άρα ο πίνακας θα έχει μέγεθος 5.
int array1[] = {2,4,8,16,32};
```

Για να μετατρέψουμε όλα τα στοιχεία ενός πίνακα με αριθμούς σε μηδενικά μπορούμε να ορίσουμε τον πίνακα ως εξής:

```
int EmptyArray[5] = { };
```

Επεξεργασία στοιχείων του πίνακα:

Για να επεξεργαστούμε ή να διαβάσουμε ένα από τα στοιχεία του πίνακα, μπορούμε να κάνουμε το εξής:

```
int array1[] = {2,4,8,16,32};  
cout<<array1[0]<<endl; // τυπώνει το πρώτο στοιχείο του πίνακα: 2  
cout<<array1[1]<<endl; // τυπώνει το δεύτερο στοιχείο του πίνακα: 4  
cout<<array1[2]<<endl; // τυπώνει το τρίτο στοιχείο του πίνακα: 8
```

Σημείωση: Όταν μετρούμε τα στοιχεία του πίνακα, ξεκινάμε από το μηδέν.

Για να επεξεργαστούμε όλα τα στοιχεία του πίνακα, μπορούμε να χρησιμοποιήσουμε μία από τις δομές επανάληψης, συνήθως την for:

```
int array1[5] = { };  
  
// Ο χρήστης ορίζει τα στοιχεία 0 μέχρι το 4:  
for(int i=0; i<5; i++){  
    cin>>array1[i];  
}
```

Συναρτήσεις:

Οι συναρτήσεις βοηθούν όταν θέλουμε να εκτελέσουμε κάποιες εντολές πάνω από μία φορά. Είναι πολύ σημαντικές αν θέλουμε να έχουμε συγυρισμένο κώδικα.

```
#include <iostream>
using namespace std;

/* Αν ο τύπος της συνάρτησης δεν είναι void, πρέπει οπωσδήποτε να
επιστρέψει μία μεταβλητή. */
int add(int a, int b){
    return a+b;
}

// Ελέγχει αν ένας αριθμός είναι ζυγός.
bool isEven(int num){
    return num%2==0;
}

/* Όσες συναρτήσεις ορίζονται μετά την main, πρέπει να οριστούν και
πριν την main με αυτόν τον τρόπο: */
void PrintSomething(string whatToPrint);

// Η main είναι η κύρια συνάρτηση. Καλείται από μόνη της.
int main()
{
    // Θα εμφανιστεί στην οθόνη η φράση "test".
    PrintSomething("test");
    cout<<add(3,5)<<endl;    // 3+5=8
    cout<<isEven(5)<<endl;  // 5
    return 0;
}

// Η void δεν επιστρέφει τίποτα. Μόνο εκτελεί εντολές
void PrintSomething(string whatToPrint){
    cout<<whatToPrint<<endl;
}
```


Αρχεία (βιβλιοθήκη fstream):

Εισαγωγή Βιβλιοθήκης:

```
#include <fstream>
```

Διαβάζοντας ένα αρχείο:

Για να διαβάσουμε ένα αρχείο, πρέπει να ορίσουμε μία εντολή τύπου ifstream*. Μπορούμε να της δώσουμε ότι όνομα θέλουμε και στην παρένθεση βάζουμε το όνομα του αρχείου που θα διαβάσουμε.

Σημείωση: Αν δεν βρεθεί το αρχείο, το πρόγραμμα θα βγάλει σφάλμα!

**ifstream: σημαίνει Input File Stream.*

```
// Όνομα Μεταβλητή Όνομα Αρχείου  
ifstream fin ("input.txt");
```

Έπειτα, μπορούμε να χρησιμοποιήσουμε την εντολή όπως χρησιμοποιούμε την εντολή cin, δηλαδή:

```
string value;  
fin>>value;
```

Κάθε φορά που εκτελούμε την εντολή fin, η fin πηγαίνει στην επόμενη γραμμή. Γι' αυτό υπάρχει και μία συνάρτηση που ελέγχει αν τελείωσαν οι γραμμές του αρχείου που διαβάζουμε, η οποία ονομάζεται eof()*

**eof(): σημαίνει End Of File.*

```
ifstream fin("input.txt");  
string value;  
  
// Τυπώνουμε στην οθόνη το περιεχόμενο του input.txt.  
while(!fin.eof()){  
    fin>>value;  
    cout<<value<<endl;  
}
```

Τέλος, όταν τελειώσουμε με την χρήση της ifstream, πρέπει να την κλείσουμε όπως πιο κάτω:

```
fin.close();
```

Επεξεργασία αρχείων:

Για να επεξεργαστούμε ένα αρχείο, πρέπει να ορίσουμε μία εντολή τύπου `ofstream*`. Μπορούμε να της δώσουμε ότι όνομα θέλουμε και στην παρένθεση βάζουμε το όνομα του αρχείου που θα επεξεργαστούμε.

Σημείωση: Αν δεν βρεθεί το αρχείο, το πρόγραμμα θα το δημιουργήσει αυτόματα!

**ofstream: σημαίνει Output File Stream.*

```
// Όνομα Μεταβλητή Όνομα Αρχείου  
ofstream fout ("output.txt");
```

Αν θέλουμε να προσθέσουμε δεδομένα στο τέλος του αρχείου, χωρίς να διαγραφούν τα προηγούμενα, πρέπει να ορίσουμε την `ofstream` λίγο διαφορετικά:

```
// ios::app σημαίνει append (δηλ. πρόσθεσε στο τέλος του αρχείου).  
ofstream fout("output.txt", ios::out | ios::app);
```

Έπειτα, μπορούμε να χρησιμοποιήσουμε την εντολή όπως χρησιμοποιούμε την εντολή `cout`, δηλαδή:

```
string value = "Hello World!";  
  
// προσθέτουμε σε καινούρια γραμμή, την φράση "Hello World!"  
ofstream fout("output.txt", ios::out | ios::app);  
fout<<endl<<value;
```

Τέλος, όταν τελειώσουμε με την χρήση της `ofstream`, πρέπει να την κλείσουμε και αυτήν όπως πιο κάτω:

```
fout.close();
```

ΤΕΛΟΣ