- www.cplusplus.com : C++ Language Tutorial
- www.learncpp.com
- www.cprogramming.com/tutorial.html
- www.tutorialspoint.com/cplusplus/cpp_tutorial.pdf
- C++ Primer, 4th Edition
- C++ How To Program, 5th Edition

C , C++

Compilers

```
gcc -o executable_file  source_file.c

g++ -o executable_file  source_file.cpp
```

Integrated Development Environment (IDE)

Microsoft Visual Studio
Code::Blocks (http://www.codeblocks.org)
Dev C++

C

```
#include <stdio.h>


int main()
{
  printf("Hello World\n");
  return 0;
}
```

C++

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

int main()
{
  cout << "Hello World"<<endl;
  cout << "Hello World\n";
  return 0;
}
```

Visual Studio

```
#include "stdafx.h"
```

precompiled headers

Χωρίς την εντολή `using namespace std;`

```
#include <iostream>

int main()
{
  std::cout << "Hello World" << std::endl;
  return 0;
}
```

Διάκριση μικρών-κεφαλαίων

```
#include <iostream>

int Main()
{
  std::cout << "Hello World" << std::endl;
  return 0;
}
```

λάθος, το όνομα `Main` δεν είναι ορισμένο

```
#include <iostream>
using namespace std;
int main()
{cout << “Hello World”<<endl; return 0;}
```

```
#include <iostream.h>
using namespace std;
int main()
{
   cout << “Hello World”
   <<endl;
   return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello
              World" <<endl;
   return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
   cout << "Hello \
            World" <<endl;
   return 0;
}
```

Συνέχεια στη επόμενη γραμμή

Είσοδος από το πληκτρολόγιο

```
#include <iostream>
using namespace std;
int main()
{
   int a,b;

   cout << "δωσε τιμη για το a";
   cin >> a;
   cout << endl;
   cout << "δωσε τιμη για το b";
   cin >> b;

   cout << " a * b = " << a*b << endl;
   return 0;
}
```

Format

I/O manipulators

```
#include <iomanip>
```

`setw(n)` : n = εύρος πεδίου εκτύπωσης. Ισχύει για την επόμενη εκτύπωση

`setprecision(n)`: n = αριθμός ψηφίων. Ισχύει μέχρι να αλλάξει το n

Output format

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
   int a;
   float x;
   a = 10;
   x = 125258.73
   cout << setw(5) << a << endl;    ___10


   cout << setprecision(3) << x;    1.25e+005
   cout << setprecision(8) << x;    125258.73


   return 0;
}
```

Σχόλια (comments)

```
#include <iostream>
using namespace std;
int main()  // program start
{
   float x; // τι είναι η x ?
   cout << "Hello World" << /* new line */ endl;
   return 0;
}
```

// Σχόλιο μιας γραμμής

/* αρχή σχολίου
.....
.....
..... τέλος σχολίου */

C Preprocessor

Οδηγίες (Directives)

```
#include

#define

#undef

#ifdef , #ifndef

#if defined  , #if !defined

#if , #elif , #else , #endif
```

Οδηγίες (directives)

- `#include`
  - `#include <iostream>`
  - `#include "header1.h"`
- `#define` *αναγνωριστικό τιμή* (Macro defines , macro constants)
  - `#define PI 3.14159265`
  - `#define PI 4 * atan(1)`

```
x = 2 * PI * r  ⟶  x = 2 * 3.14159625 * r
```

  - `#define REAL double`
  - `#define MY_NAME`
- `#undef`
  - `#undef REAL`
  - `#undef MY_NAME`

Αρχεία επικεφαλίδων - Header files

- Αρχεία χωρίς κατάληξη `.h`
  - `#include <iostream>`
  - `#include <string>`
  - Μετά την καθιέρωση των ANSI standards για τη C++

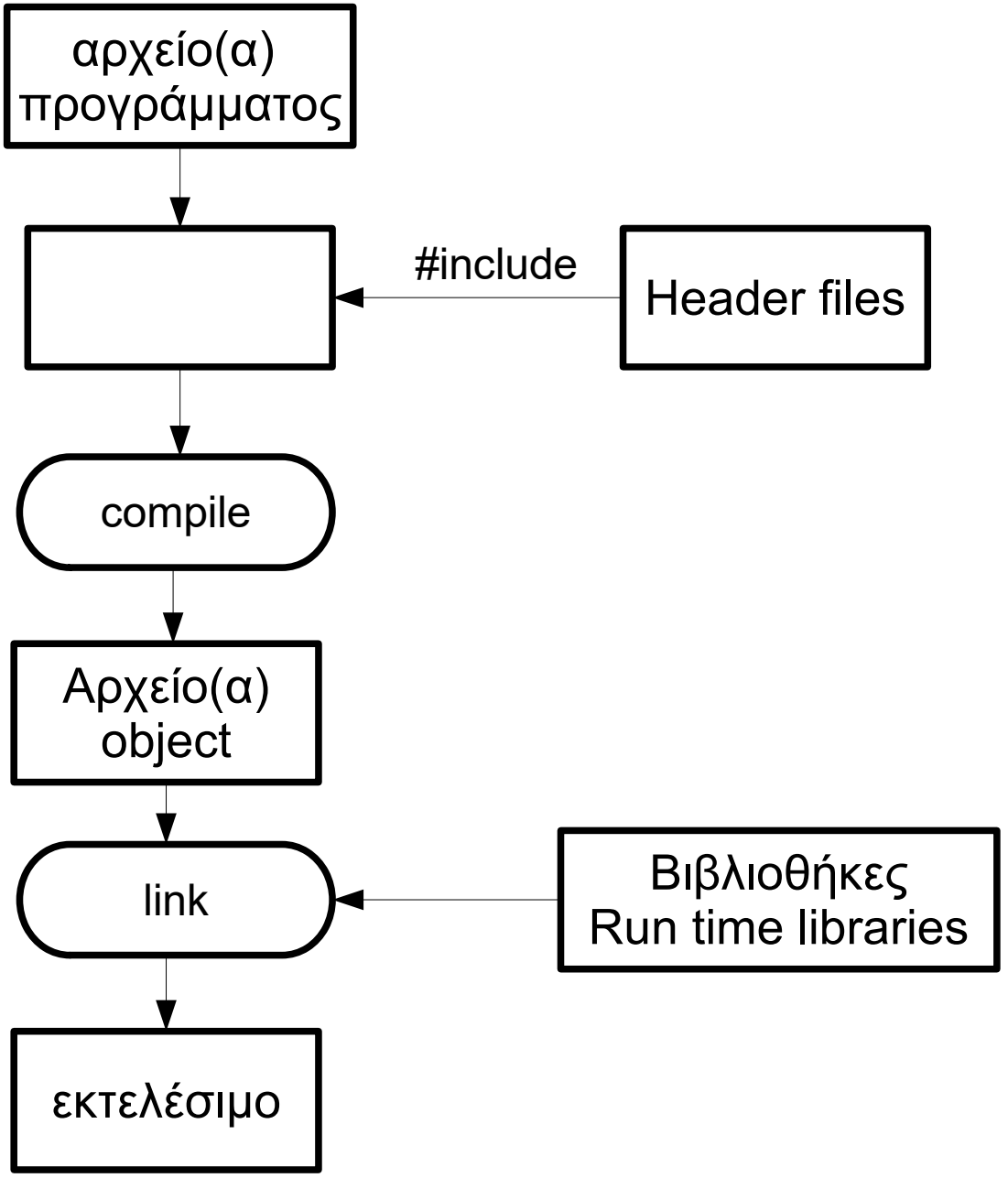
- Αρχεία με καταληξη `.h`
  - `#include <iostream.h>` (για παλαιότερους compilers)
  - `#include <string.h>`
  - Διατηρούν τη συμβατότητα με παλαιότερα προγράμματα

Μεταγλώττιση υπό συνθήκες (conditional compilation)

```
#if defined CONDITION
// κώδικας C++
#endif
```

```
#if defined WINDOWS
// κώδικας για Windows
#endif
#if defined LINUX
// κώδικας για Linux
#endif
```

```
#define C_compiler
//#define Cpp_compiler


#ifdef C_compiler
  #include <stdio.h>  //inclue C libraries
#endif

#ifdef Cpp_compiler
  #include <iostream> //inclue C++ libraries
  using namespace std;
#endif


int main()
{
  #ifdef C_compiler
    printf("This is a C programm \n");
  #endif

  #ifdef Cpp_compiler
    cout << "This is a C++ programm " << endl;
  #endif

  return 0;
}
```


αρχείο(α)
προγράμματος
#include
Header files
compile
Αρχείο(α)
object
link
Βιβλιοθήκες
Run time libraries
εκτελέσιμο

```
#include <iostream>
using namespace.std
int main()
{
    float x,y;
    cin >> x;
    cin >> y;
    // cin >> x >> y;
    cout << "Η μέση τιμή των x,y είναι" << mean_value(x,y);
    cout << endl;


float mean_value(float a, float b)
{
    return (a + b)/2;
}
```

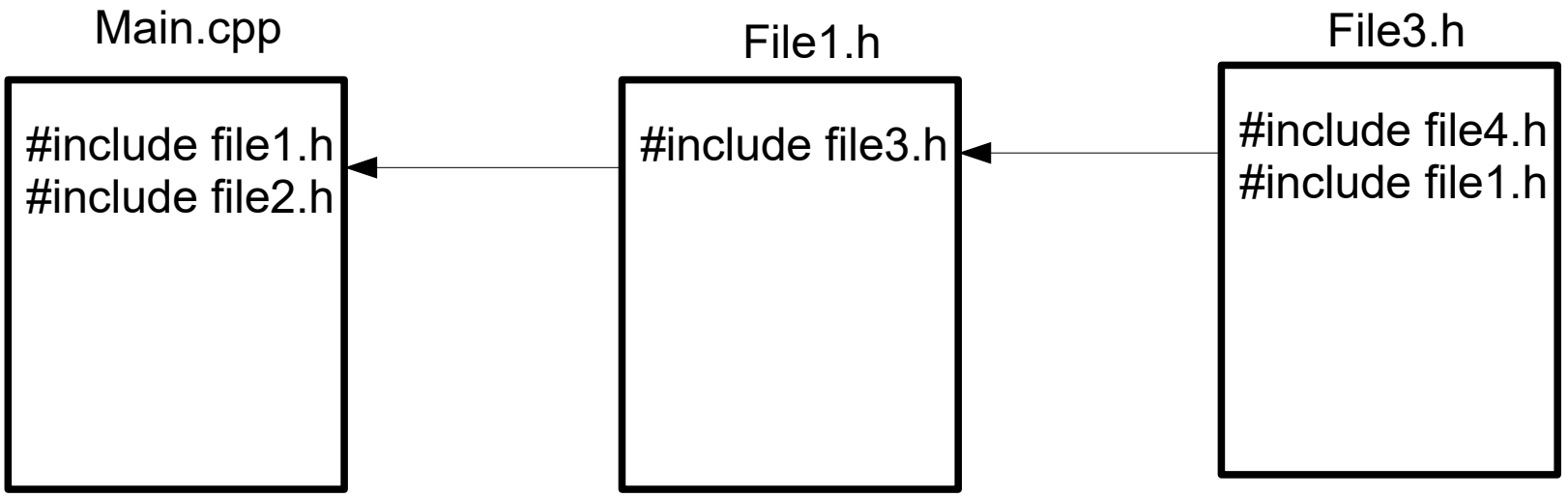
```
#include <iostream>
using namespace.std
int main()
{
    float x,y;
    cin >> x;
    cin >> y;
    // cin >> x >> y;
    cout << "Η μέση τιμή των x,y είναι" << mean_value(x,y);
    cout << endl;
}

float mean_value(float a, float b)
{
    return (a + b)/2;
}
```

//αρχείο example1.cpp

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

float mean_value(float a, float b);
//float mean_value(float , float );


int main()
{
    float x,y;
    cin >> x;
    cin >> y;
    // cin >> x >> y;
    cout << "Η μέση τιμή των x,y είναι" << mean_value(x,y);
    cout << endl;
    return 0;
}
```

//αρχείο my_functions.cpp

```
float mean_value(float a, float b)
{
    return (a + b)/2;
}
```

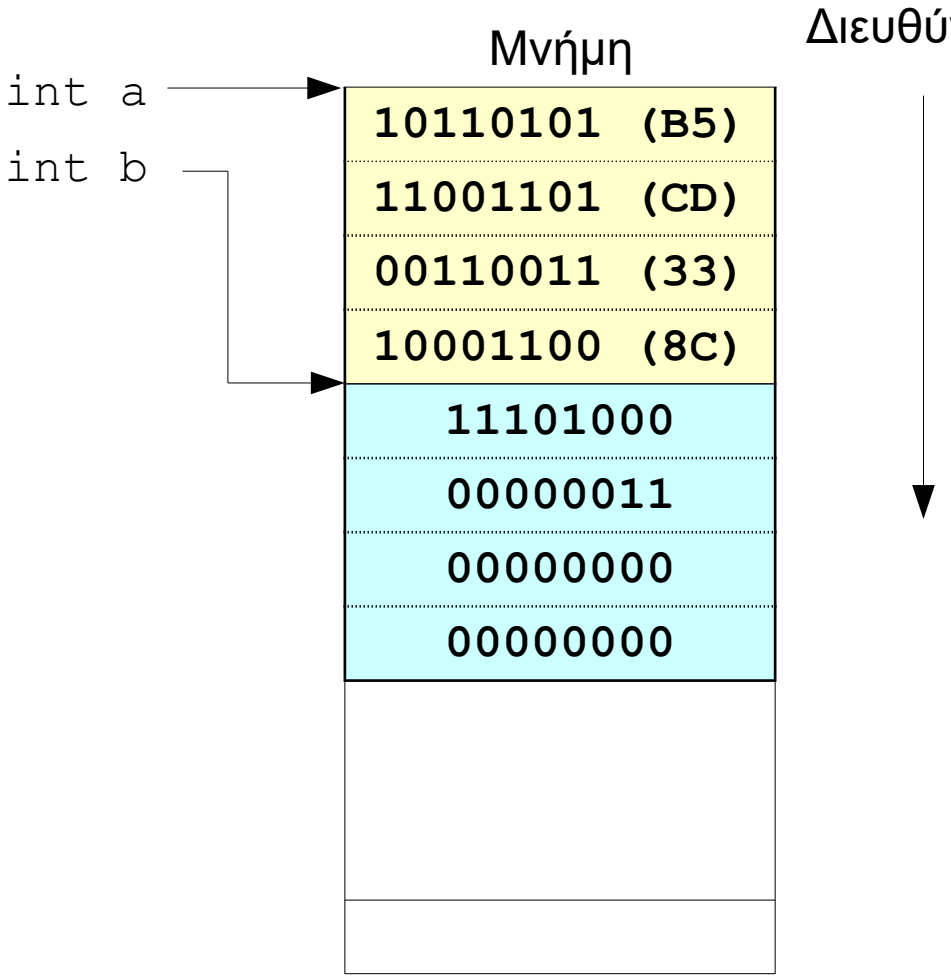

Main.cpp
#include file1.h
#include file2.h
File1.h
#include file3.h
File3.h
#include file4.h
#include file1.h

```
//αρχείο my_functions.h
//header guard
#ifndef MY_FUNCTIONS
#define MY_FUNCTIONS

float mean_value(float x, float y);

#endif
```

//αρχείο example1.cpp

```
#include <iostream>
#include “my_functions.h”
using namespace.std


int main()
{
    float x,y;
    cin >> x;
    cin >> y;
    // cin >> x >> y;
    cout << “Η μέση τιμή των x,y είναι” << mean_value(x,y);
    cout << endl;
    return 0;
}
```

Δήλωση μεταβλητών



8C33CDB5 hex
2352205237 dec

Αρχικές τιμές

```
int main()
{
   int b = 1000;
   int b(1000);

}
```

Ακέραιος 2 bytes (16 bit)

Δεκαεξαδικό	FFFC
Δυαδικό	1111 1111 1111 1100
Δεκαδικός χωρίς πρόσημο	65532
Δεκαδικός με πρόσημο	-4

Τύποι στοιχείων

- Λογικές μεταβλητές
  - `bool`
- Χαρακτήρες
  - `[signed] char`
  - `unsigned char`
- Ακέραιοι
  - `[signed] int`
  - `unsigned int`
  - `[signed] short` (`[signed] short int`)
  - `unsigned short` (`unsigned short int`)
  - `[signed] long` (`[signed[ long int`)
  - `unsigned long` (`unsigned long int`)
  - `[signed] long long` (`signed long long int`)
  - `[unsigned] long long` (`unsigned long long int`)
- Πραγματικοί (κινητής υποδιαστολής)
  - `float`
  - `double`
  - `long double`

Χαρακτήρες

```
char c1,c2;
c1='A';
c2=65;

c1=c1+('a'-'A');
```

Μη εκτυπώσιμοι χαρακτήρες
Χαρακτήρες ελέγχου

	Octal	**hex**	
\n	\012	\0xA	new line
\t	\011	\0x9	horizontal tab
\v	\013	\0xB	vertical tab
\f	\014	\0xC	form feed
\b	\010	\0x8	back space
\a	\007	\0x7	bell
\r	\015	\0xD	carriage return

32		52	4	72	H	92	\	112	p
33	!	53	5	73	I	93	]	113	q
34	“	54	6	74	J	94	^	114	r
35	#	55	7	75	K	95	_	115	s
36	$	56	8	76	L	96	`	116	t
37	%	57	9	77	M	97	a	117	u
38	&	58	;	78	N	98	b	118	v
39	'	59	:	79	O	99	c	119	w
40	(	60	<	80	P	100	d	120	x
41	)	61	=	81	Q	101	e	121	y
42	*	62	>	82	R	102	f	122	z
43	+	63	?	83	S	103	g	123	{
44	,	64	@	84	T	104	h	124	|
45	-	65	A	85	U	105	i	125	}
46	.	66	B	86	V	106	j	126	~
47	/	67	C	87	W	107	k	127	
48	0	68	D	88	X	108	l	128	
49	1	69	E	89	Y	109	m	129	
50	2	70	F	90	Z	110	n	130	
51	3	71	G	91	[	111	o	131	

Ακέραιοι

1 byte signed	-127	127
1 byte unsigned	0	255
2 bytes signed	-32768	32767
2 bytes unsigned	0	65535
4 bytes signed	-2,147,483,648	2,147,483,647
4 bytes unsigned	0	4,294,967,296
8 bytes signed	-9,223,372,036,854,775,807	9,223,372,036,854,775,807
8 bytes unsigned	0	18,446,744,073,709,551,615

```
short int a;
a = 32765;
a = a + 5;
cout << a;
```
-32766

```
unsigned short b;
b = 65534;
b = b + 2;
cout << b;
```
0

Πραγματικοί

	Ubuntu 64bit G++	Windows 64 bit Visual C++ v10 32bit	Windows 64 bit g++ 32bit
Boolean	1 Bytes	1 Bytes	1 Bytes
Character	1 Bytes	1 Bytes	1 Bytes
Wide Character	4 Bytes	2 Bytes	2 Bytes
Short Integer	2 Bytes	2 Bytes	2 Bytes
Integer	4 Bytes	4 Bytes	4 Bytes
Long Integer	8 Bytes	4 Bytes	4 Bytes
Long Long Integer	8 Bytes	8 Bytes	8 Bytes
Float	4 Bytes	4 Bytes	4 Bytes
Double	8 Bytes	8 Bytes	8 Bytes
Long Double	16 Bytes	8 Bytes	12 Bytes

float	4	3.4 e ± 38
double	8	1.7 e ± 308
Long doule	12	

Σταθερές

- `20  int`
- `20u  unsigned int`
- `20l  long`
- `20ul unsigned long`
- `0115 οκταδικό`
- `0x1A δεκαεξαδικό`
- `0b10101010 δυαδικό (gcc)`
- `3.14159`
- `3.14159L long double`

Σταθερές

```
#define ELECTRON_CHARGE 1.6e-19
```

- Δεν είναι μεταβλητή, δεν αποθηκεύεται στη μνήμη
- Ορίζεται πριν από τις συναρτήσεις (και τη `main` αν είναι στο ίδιο αρχείο)
- Εμβέλεια: Όλο το αρχείο (global)

```
const double ElectronCharge = 1.6e-19;
```

- Αποθηκεύεται στη μνήμη όπως οι μεταβλητές
- Η τιμή δεν μπορεί να αλλάξει. (read only)
  - Η εντολή `ElectronCharge = 100;` θα δώσει σφάλμα μεταγλώττισης
- Ορίζεται οπουδήποτε μέσα στο πρόγραμμα όπως και οι άλλες μεταβλητές
- Η εμβέλεια της είναι τοπική ή καθολική

Μετατροπές τύπων-Αυτόματες μετατροπές

```
float x;
int y;
double z;
z=x+y;//οι x και y πρέπει να είναι του ίδιου τύπου. Η y μετατρέπεται σε float
        και το αποτέλεσμα μετατρέπεται σε double
```

- Σειρά μετατροπής
  long double>double>float>unsigned long>long>unsigned int>int

```
y=z+x;//Η y μετατρέπεται σε double και το αποτέλεσμα στρογγυλοποιείται
        σε int
```

- Microsoft Visual Studio
  warning C4244: '=' : conversion from 'double' to 'int', possible loss of data

```
x=z+y;//
```

warning C4244: '=' : conversion from 'double' to 'float', possible loss of data

```
x=5/3;   // x=1
```

Μετατροπές τύπων - casting

```
float x;
int y;
double z;
z=(double)x+(double)y;
```
μετατροπή στη C

Μετατροπή στη C++

```
static_cast<double> x
```

Αριθμητικοί τελεστές – προτεραιότητες
+ - * / % ++ -- = ,

- Πράξεις ακεραίων

```
int a=3,b=5,c;
c = a/b;
```
(αποτέλεσμα 0)

- a++ ++a

```
a = 2
b = a++ + 1;
c = ++a + 1;
```
b = 3 , a = 3

a = 4 , c = 5

- a=10+(b=3)

```
b=3
a=10+b
```

- a += b

```
a=a+b
```

- a = (b=1,b+5)

```
b=1 ; a=b+5 ;
```

- z = (x > y) ? x : y;

```
if (x>y) z=x else z=y;
```

Αποτελέσματα του τελεστή ++

```
int y = 5;
int x ;
x = ++y + ++y;
```

```
++y          y = 6
++y          y = 7
             x = 7 + 7 = 14
```

```
int y = 5;
int x ;
x = ++y + ++y + 1;
```

```
++y          y = 6
++y + 1      y = 7   ++y+1=8
x = 8 + 7    x = 15
```

```
int y = 5;
int x ;
x = y++ + ++y;
```

```
y++          y = 5
++y          y = 6
             x = 6 + 6
y++          y = 7
```

```
int y = 5;
int x ;
x = y++ + ++y + 1;
```

```
y++          y = 5
++y + 1      y = 6 ++y+1=7
x = 6 + 7    x= 13
             y = 7
```

```
int y = 5;
int x ;
x = ++y + 1 + ++y;
```

Microsoft VS 2010 x=15 y=7
Code::Blocks 12.11 x=14 y=7

```
x = ++y + (1 + ++y)
```

```
int main() //Gauss method for Easter date computation
{
  int year, start_year,end_year, a, b, c, d, e;
  cout << "Give Start Year : ";
  cin >> start_year; cout << endl;
  cout << "Give End Year   : ";
  cin >> end_year; cout << endl;
  for (year = start_year ; year <= end_year ; year++)
  {
    a = year % 19;
    b = year % 4;
    c = year % 7;
    d = (19*a+15) % 30;
    e = (2*b+4*c+6*d+6) % 7;
    cout << "Easter in the year " << year;
    if (d+e+4 > 30) // Easter in May
       cout << "   May    " << d+e-26 << endl;
    else // Easter in April
       cout << "   April  " << d+e+4 << endl;
  }
   return 0;
}
```

bitwise

- `&` και (and)

```
  11000011
& 10001001
  10000001
```

- `|` ή (or)

```
  11000011
| 10001001
  11001011
```

- `^` αποκλειστικό ή (xor)

```
  11000011
^ 10001001
  01001010
```

- `~` συμπλήρωμα ως προς ένα

```
~01010011 = 10101100
```

- `a << n` ολίσθηση του `a` αριστερά `n` θέσεις

```
01001000 << 1  =  10010000
```
(72 << 1 =144)

- `a >> n` ολίσθηση του `a` δεξιά `n` θέσεις

```
01001000 >> 1  =  00100100
```
(72 >> 1 =36)

Τελεστές

- `unsigned short a = 0x8A (10001010,138)`
- τι τιμή έχει το bit 3; **`a & (1 << 3)`**

```
                10001010            10000010
&(AND)          00001000          & 00001000
                00001000 = 8        00000000 = 0
```

- μηδενισμός του bit 4 **`a & ~(1 << 4)`**

```
                10011010
              & 11101111   ~(1<<4)= ~(00010000)=11101111
                10001010
```

- θέση (set) του bit 6 **`a | (1 << 6)`**

```
                10001010
|(OR)          01000000
                11001010
```

- αντιστροφή του bit 1 **`a ^ (1 << 1)`**

```
                10001010            10001000
^(XOR)          00000010          ^ 00000010
                10001000            10001010
```

Σχεσιακοί τελεστές (==, !=, >, <, >=, <=)
Λογικοί Τελεστές (!, &&, ||)

```
( a >= 5) && (a <= 10)        // 5 ≤ a ≤ 10
( a < 5) || ( a > 10 )        // 5 > a > 10
!(( a >= 5) && (a <= 10))     // 5 > a > 10
```

Μια αριθμητική μεταβλητή έχει και λογική τιμή: a = 0 false , a ≠ 0 true

```
a=1 , b=10 : a == b → false (0)   !(a == b) → true (≠0)

    (!a == b) ↔ (!a)==b ↔ (!1==10) ↔ 0==10 → false

if (a=1) → true
  {  }


float a,b :   a == b ?
```

Παρενθέσεις

```
  ( ( a+b ) * ( ( m+n ) / ( f+g ) * k )+w ) / ( q+p )
0 1 2     1   2 3     2   3     2     1   0   1     0
```

$$\frac{(a+b)\frac{m+n}{f+g}k+w}{q+p}$$

```
   ( ( a+b ) * ( ( m+n ) / ( f+g ) * k )+w  / ( q+p ) )
 0 1 2     1   2 3     2   3     2     1      2     1 0
```

$$(a+b)\frac{m+n}{f+g}k+\frac{w}{q+p}$$

External variables

```
//File_1.cpp
extern int m
int main()
{
  m = .....
}
```

```
//File_2.cpp
int m;
function_1(){}
function_2(){}
```

Εμβέλεια μεταβλητών

```
extern float e_x;
float g_x;
float mean(float ,float );

int main()
{
  float x,y,z,t;
  z= mean(x,y);
  {
    float w;
    w=z*z;
    g_x=w;
  }
  cout << g_x;
  cout << w;
}

float mean(float a,float b)
{
  float x;
  x = (a+b)/2;
  return x;
}
```

Η δήλωση υπάρχει σε άλλο αρχείο

global

Τοπικές στη main

Τοπική στο block

Άγνωστη εκτός το block (out of scope)

Τοπική στη mean

Εντολές διακλάδωσης

```
if (λογική παράσταση)
   εντολή;
else
   εντολή;
```

```
if (λογική παράσταση)
{
   εντολή_1;
   εντολή_2;
   ........
}
else
{
   εντολή_3;
   εντολή_4;
   .........
}
```

```
if (λογική παράσταση)
{
   εντολή;
   ........
}
else if
{
   εντολή;
   .........
}
else if
{
   εντολή;
   .........
}
else
{
   εντολή;
   .........
}
```

if

```
int main()
{
    char c;
    cout << “Enter a character : ”;
    cin >> c;
    if ( c >= 'a' && c <= 'z')
       c = c – ('a' -'A');
}
```

```
if ( c >= 'a' && c <= 'z');
   c = c – ('a' -'A');
```

```
if ( c >= 'a' && c <= 'z')
            ;
c = c – ('a' -'A');
```

Κενή εντολή

if

```
if ( x > 0)
   if ( x>=1 && x<= 10)
      // 1≤x≤10
   else
      // 0<x<1 ή x>10
```

Εντολή που εκτελείται αν x>0

```
if ( x > 0)
   {
    if ( x>=1 && x<= 10)
      // 1≤x≤10
   }
else
   // x ≤ 0
```

```
if ( x > 0)
{
   if ( x>=1 && x<= 10)
      // 1≤x≤10
   else
      // 0<x<1 ή x>10
}
```

if

```
int a;
float b,c;
cin >> a >> b;
if ( a == 3)
  cout <<"division by zero";
else
  c=b/(a-3);
```

```
if ( a == 0)
  cout <<"division by zero";
else
  c=b/a;
```

```
if ( a != 3)
  c=b/(a-3)
else
  cout <<"division by zero";
```

```
if ( a = 3)
   cout << "division by zero";
else
    c=b/(a-3);
```

Πάντα true

```
if ( a = 0)
  cout << "division by zero";
else
  c=b/a;
```

Πάντα false

```
if ( a != 0)
  c=b/a;
else
  cout <<"division by zero";
```

if - else if

```
int main()
{
   double x,y,z;
   char oper;
   bool success=true;
   cout << "Enter operand1 operator operand2 :";
   cin >> x >> oper >> y;
   if (oper == '+')
      z = x+y;
   else
      if (oper == '-')
         z = x-y;
      else
         if (oper == '*')
            z = x*y;
         else
            if (oper == '/' && y != 0)
               z = x/y;
            else
               success = false;

   if (success)
       cout << x  << oper << y << " = " << z << endl;

   return !success;
}
```

switch

```
int main()
{
   double x,y,z;
   char oper;
   bool success=true;
   cout << "Enter operand1 operator operand2 :";
   cin >> x >> oper >> y;
   switch (oper)
   {
      case '+':
         z = x+y;
         break;
      case '-':
         z = x-y;
         break;
      case '*':
         z = x*y;
         break;
      case '/':
         if (y != 0) z = x/y;else success = false;
         break;
     default:
         success = false;
   }
   if (success)
       cout << x  << oper << y << " = " << z << endl;

   return success;
}
```

Μια τιμή, όχι περιοχή ή σύνολο τιμών

switch

```
int main()
{
   double x,y,z;
   char oper;
   bool success=true;
   cout << “Enter operand1 operator operand2 :”;
   cin >> x >> oper >> y;
   switch (oper)
   {
      case '+':
         z = x+y;
         break;
      case '-':
         z = x-y;
         break;
      case '*':
         z = x*y;
         break;
      case '/':
         if (y != 0) z = x/y;
      default:
         success = false;
   }
   if (success)
       cout << x  << oper << y << " = " << z << endl;

   return success;
}
```

switch

```
int main()
{
 char c;
 cin >> c
 switch (c)
 {
  case 'A':
  case 'a':
  case 'E':
  case 'e':
  case 'O':
  case 'o':
  case 'I':
  case 'i':
  case 'U':
  case 'u':
    cout<<"Character is vowel";
    break;
  case 'Y':
  case 'y':
    cout<<"Character is vowel
           or consonant";
    break;
  case 'B':
  case 'B':
  case 'C':
  case 'c':
  case 'D':
  case 'd':
  case 'F':
  case 'f':
  ........
  ........
  case 'Z':
  case 'z':
    cout<<"Character is consonant";
    break;
  default:
    cout <<"Other character";
 }
 return 0;
}
```

Εντολή διακλάδωσης χωρίς συνθήκη
goto

```
int a;
char c;
cout << “Enter a digit”;
try_again:
cin >> c;
if ( !(c >= '0' && c <= '9') )
{
  cout << “Wrong Character, Try Again”;
  goto try_again;
}
a = c – '0';
```

Ετικέτα (label)

Η εντολή goto μπορεί να αντικατασταθεί με μια εντολή επανάληψης

Εντολές επανάληψης
while , do-while

```
int a;
char c;
cout << “Enter a digit”;
cin >> c;
while ( !(c >= '0' && c <= '9') )
{
  cout << “Wrong Character, Try Again”;
  cin >> c;
}
a = c – '0';
```

```
do
{
  cout << “Enter a Digit”;
  cin >> c;
}while ( !(c >= '0' && c <= '9') );
a = c – '0';
```

Εντολές επανάληψης
while , do-while

```
while(παράσταση)
{
   //εκτελούνται όταν η παράσταση δεν είναι μηδέν (true)
}
```

```
while(παράσταση);   //κενή εντολή
```

```
while(1)
{      }          //endless loop
```

while

```
int main()
{
  double sumx = 0.0;
  int n = 0;
  int first,last,temp,i;
  cout << "Enter the first integer :";
  cin >> first; cout << endl;
  cout << "Enter the last integer :";
  cin >> last; cout << endl;
  if (first>last)
  {
    temp = first;    // αν first > last
    first = last;    // αντιμετάθεση των τιμών των μεταβλητών
    last = temp;
  }
  i = first;
  while(i <= last)
  {
    n++;       // πλήθος αριθμών του αθροίσματος
    sumx += (double)i++;
  }
  cout << "Sum of integers from " << first << " to "<<last
       << " = " << sumx << endl;
  cout << "Average value = " << sumx/n;  //sumx/(last-first)
  return 0;
}
```

Εντολή επαναλήψεων `for` (1)

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
  double x = 0.0;
  for (int i = 0; i <= 100; i++)
    x = x + (double)i;
  cout << "Sum = " << x << endl;
  cout << "Mean = " << x / i << endl; // i τοπική στη for
  return 0;
}
```

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
  double x = 0.0;
  for (int i = 0; i <= 100; ) // δεν υπάρχει μεταβολή του i
    x = x + (double)i++;      //  το i αυξάνεται εδώ
  cout << "Sum = " << x << endl;
  return 0;
}
```

Εντολή επαναλήψεων `for` (2)

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{
  double x = 0.0;
  int i,k;
  for ( i = 1 , k = 1; i <= 100; i++ , k = k + 2 )
    x = (double)i / (double)k;
  cout << "x = " << x << endl;
  return 0;
}
```

```
int i;
cout << " Enter starting index "
cin >> i;                    // αρχική τιμή του i εδώ
for (  ; i <= 1000; i++)     // η θέση στη for για την αρχική τιμή μένει κενή
{     }

for ( int i = 0 ; i <= 1000; i++) // καθυστέρηση
            ;
```

```
for ( ; ; )  endless loop
{  }
```

```
int a;
cout << " Enter a ( ≠ 0 ) : ";
cin >> a;
cout << endl;

for ( ; a == 0 ; )
{
  cout << " a = 0 , Enter a : ";
  cin >> a;
  cout << endl;

}
```

Παράληψη επαναλήψεων
Εντολή `continue`

```
#include <iostream>

using namespace std;
int main()
{
  for (int i = 0; i <= 100; i++)
  {
    if ( !(i % 2) )      //i%2=0 false. !0 true
      continue;
    cout << i << endl;   //τυπώνει τους μονούς αριθμούς
  }
}
```

Διακοπή επαναλήψεων `break`

```
#include <iostream>
using namespace std;
int main()
{
  int n;
  cout << "Enter an integer :";
  cin >> n;
  cout << endl;
  unsigned long long factorial_i = 1;
  double factorial_d = (double)1;
  for (int i = 1; ; i++)
  {
    if (i > n)
      break;
    factorial_d *= (double)i;
    factorial_i *= i;
  }
  cout << n << "! = " << factorial_i << endl;
  cout << n << "! = " << factorial_d << endl;
  return 0;
}
```

$$\frac{\pi}{2}=\frac{2}{1}\frac{2}{3}\frac{4}{3}\frac{4}{5}\frac{6}{5}\frac{6}{7}\frac{8}{7}\frac{8}{9}\cdots$$

```
int main()
{
    double product1 = (double)1;
    double product2 = (double)1;
    double x;

    for (int i=1;i < 2000;i++)
    {
        x = (double)(2*i);
        product1 *= (x/(x-1))*(x/(x+1));
    }
    for (int i=2;i < 2000;i += 2)
    {
        x = (double)i;
        product2 *= (x/(x-1))*(x/(x+1));
    }
    cout << "Product = "<<product1 << endl;
    cout << "Pi = "<< product1 * 2 << endl;
    return 0;
}
```

$$\frac{\pi}{2}=\prod_{i=1}^{n}\frac{2\mathrm{i}}{2\mathrm{i}-1}\frac{2\mathrm{i}}{2\mathrm{i}+1}$$

$$\frac{\pi}{2}=\prod_{i=2(2)}^{i}\frac{i}{i-1}\frac{i}{i+1}$$

for

$$\frac{1}{2\cdot3\cdot4}-\frac{1}{4\cdot5\cdot6}+\frac{1}{6\cdot7\cdot8}-\frac{1}{8\cdot9\cdot10}+\ldots \qquad \sum_{i=1}^{n}\frac{(-1)^{i+1}}{(2i)(2i+1)(2i+2)}$$

```
int main()
{
   double sum = (double)0.0;
   double x,current_term;
   int i=1,sign;
   for (int i=1;i < 2000;i++)
   {
      if ((i %2)== 0 ) //  αν i ζυγό  i+1 μονό
            sign = -1;
      else
            sign = 1;
      x = (double)(2*i);
      current_term = 1.0/(x*(x+1)*(x+2));
      sum += (double)sign * current_term;
   }
   cout << "sum = "<<sum << endl;
   return 0;
}
```

παρανομαστής

$$\frac{1}{2\cdot3\cdot4}-\frac{1}{4\cdot5\cdot6}+\frac{1}{6\cdot7\cdot8}-\frac{1}{8\cdot9\cdot10}+\ldots \qquad \sum_{i=1}^{n}\frac{(-1)^{i+1}}{(2i)(2i+1)(2i+2)}$$

```
int main()
{
   double sum = (double)0.0;
   double x,current_term;
   int i=1,sign;
   for (int i=1;i < 2000;i++)
   {
      if ((i %2) == 0 )
           sign = -1;
      else
           sign = 1;
      x = (double)(2*i);
      current_term = 1.0;
      for (int j=0;j<=2;j++)
         current_term *= 1.0/(x+j);
      sum += (double)sign*current_term;
   }
   cout << "sum = "<<sum << endl;
   return 0;
}
```

Υπολογισμός παρανομαστή
Με επαναλήψεις

$$\frac{1}{2\cdot3\cdot4}-\frac{1}{4\cdot5\cdot6}+\frac{1}{6\cdot7\cdot8}-\frac{1}{8\cdot9\cdot10}+\ldots \qquad \sum_{i=1}^{n}\frac{(-1)^{i+1}}{(2i)(2i+1)(2i+2)}$$

```
int main()
{
   double sum = (double)0.0;
   double x,current_term;
   int i=1,sign;
   const double threshold = 1.0e-10;
   for (int i=1;i < 2000;i++)
   {
      if ((i %2) == 0 )
           sign = -1;
      else
           sign = 1;
      x = (double)(2*i);
      current_term = 1.0;
      for (int j=0;j<=2;j++)
         current_term *= 1.0/(x+j);
      if (current_term < threshold)
          break;
      sum += (double)sign*current_term;
   }
   cout << "sum = "<<sum << endl;
   return 0;
}
```

Όριο διακοπής επαναλήψεων

while

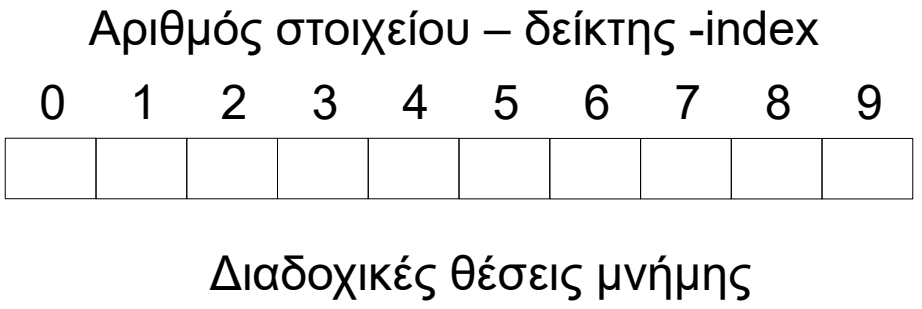
$$\frac{1}{2\cdot3\cdot4}-\frac{1}{4\cdot5\cdot6}+\frac{1}{6\cdot7\cdot8}-\frac{1}{8\cdot9\cdot10}+\ldots \qquad \sum_{i=1}^{n}\frac{(-1)^{i+1}}{(2i)(2i+1)(2i+2)}$$

```
int main()
{
   double sum = (double)0.0;
   double x,current_term;
   int i=1,sign;
   const double threshold = 1.0e-10;
   do
   {
      if ((i %2) == 0 )
           sign = -1;
      else
           sign = 1;
      x = (double)(2*i);
      current_term = 1.0/(x*(x+1)*(x+2));
      sum += (double)sign*current_term;
      i++;
   }while (current_term > threshold);

   cout << "sum = "<<sum << endl;
   return 0;
}
```

Πίνακες - Arrays

Δήλωση μονοδιάστατου πίνακα 10 ακεραίων

```
#define M 10

const int n=10;
int k = 10

int a[10];
int b[n];
int c[M];
int d[k]; //δεν επιτρέπεται γιατί k δεν είναι σταθερό
```



Αρχικές τιμές

```
int a[8] = { 1 , 5 , 21 , 45 , 32 , 78 , 66 , 9 }
```

```
int a[8] = { 1 , 5 , 21 , 45 , 32 , 78 , 66 , 9 , 3}
Error : too many initializers
```

δήλωση πίνακα 12 στοιχείων

```
double x[] = {1.1 , 5.6 , 2.3 , 4.7 , 10.5 , 8.4 ,
              7.7 , 9.6 , 21.8 , 25.5 , 33.1 , 4.8}
```

```
int a[8] = { 0 }   //όλα τα στοιχεία του πίνακα 0
```

Η C++ δεν ελέγχει τα όρια των πινάκων

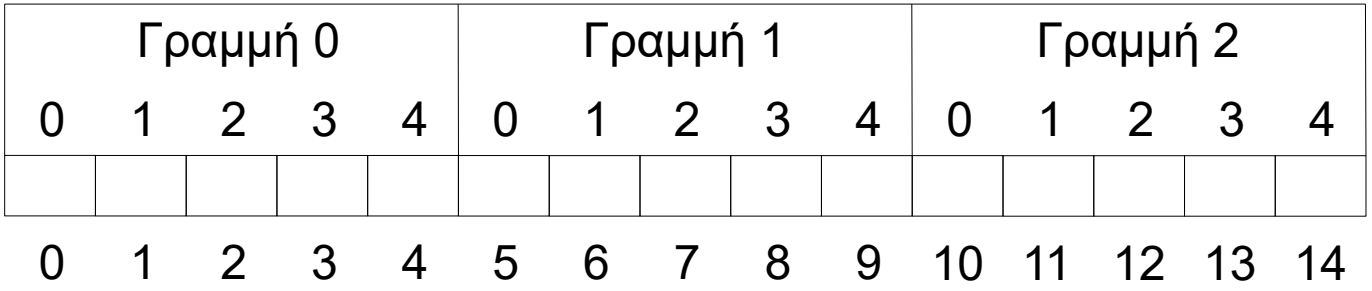
Ένα λάθος πρόγραμμα!!

```
int main()
{
  int a[10],i;
  for(i=0;i<20;i++)
     a[i]=i;
  return 0;
}
```

Το πρόγραμμα θα γράψει σε περιοχή της μνήμης που χρησιμοποιείται από άλλες μεταβλητές ή κώδικα

Πίνακες δύο ή περισσοτέρων διαστάσεων

```
const int row=3,col=5;
int x[row][col];
```



Διαδοχικές θέσεις μνήμης

```
int x[3][5]={{1,2,8,33,51},{4,12,3,21,9},{7,5,2,45,6}}

int x[ ][5]={{1,2,8,33,51},{4,12,3,21,9},{7,5,2,45,6}}
```

Εισαγωγή στοιχείων πίνακα

```
#include <iostream>
#include <math.h>

using namespace std;

int main()
{
  const int MAX=100;
  double x[MAX];
  int n;
  do
  {
    cout >> “Enter number of data points [2 to “ << MAX << “] : ”;
    cin >> n;
    cout << endl;
  }while ( (n < 2) || (n > 100) );

  for (int i=0 ; i < n ; i++)
  {
    cout << "x["<< i+1 << "] = ";
    cin >> x[i];
    cout << endl;
  }
```

Υπολογισμός διαφόρων μεγεθών

```
double min_value=x[0],max_value=x[0],mean_value=0.0;
for(int j=0 ; j < n ;j++)
{
  if (x[j] < min_value) min_value = x[j];
  if (x[j] > max_value) max_value = x[j];
  // min_value = (min_value < x[j])? min_value: x[j];
  // max_value = (max_value > x[j])? max_value: x[j];
  mean_value += x[j];
}
mean_value = mean_value/n;

double standard_deviation = 0.0;
for(int j=0 ; j < n ; j++)
   standard_deviation += pow(x[j]-mean_value,2);
standard_deviation = sqrt(standard_deviation/n);

cout << "Minimun = " << min_value << endl;
cout << "Maximum = " << max_value << endl;
cout << "Mean value = " << mean_value << endl;
cout << "Standard deviation = " << standard_deviation << endl;
```

Ταξινόμηση των στοιχείων του πίνακα

```
//αλγόριθμος COMB
const float shrink = 1.3f;
int offset;
double temp;
bool inOrder;

offset = n;
do
{
  offset = (int)(offset/shrink);
  offset = (offset == 0) ? 1 : offset;
  inOrder = true;
  for(int i=0,j=offset;i<(n-offset);i++,j++)
    if (x[i] > x[j])
    {
      inOrder = false;
      temp = x[i];
      x[i] = x[j];
      x[j] = temp;
    }
}while (!(offset == 1 && inOrder == true));
for(int i=0;i<n;i++)
     cout << "x[" << i << "] = "<< x[i] << endl;

return 0;
}
```

Συντελεστής μείωσης απόστασης

Αρχική τιμή απόστασης

Σε κάθε βήμα η απόσταση διαιρείται με τον συντελεστή

Σύγκριση του στοιχείου i με το στοιχείο j σε “απόσταση” offset

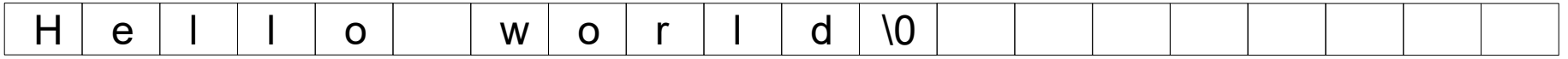
Αντιμετάθεση στοιχείων. Το μικρότερο στη θέση i

Πίνακες χαρακτήρων (της C)

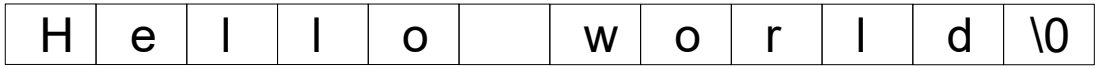
```
char my_string[20];
```

```
char message[20] = "Hello world";
```

Null character- ASCII code 0
(Τον προσθέτει ο compiler)




```
char message[] = "Hello world";
```




```
char message[]={'H','e','l','l','o',' 
',' ','w','o','r','l','d','\0'};
```

Πίνακες χαρακτήρων (της C)

```
include <iostream>
include <string>

#include <cstdio>

using namespace.std;

int main()
{
  int m;
  char first_name[20], last_name[20], full_name[50];
  cout << "Enter your first name ";
  cin >> first_name; cout << endl;
  cin >> last_name; cout << endl;

  cout << "Enter your full_name" ;
  gets(full_name);

  return 0;
}
```

Διαβάζει μέχρι το πρώτο κενό

Διαβάζει μια γραμμή μαζί με τα κενά

Δεν ελέγχεται το πλήθος των χαρακτήρων

Τυπικές συναρτήσεις για πίνακες χαρακτήρων

`strlen(str)`	Επιστρέφει το μήκος (πλήθος χαρακτήρων)
`strcpy(s1,s2)`	Αντιγράφει το στοιχεία του s2 στον s1 Πρέπει η διάσταση του s1 να ίση ή μεγαλύτερη της διάστασης του s2.
`strcmp(s1,s2)`	Συγκρίνει του δύο πίνακες και επιστρέφει: 0 αν είναι ίσοι Θετικό αριθμό αν `s1 > s2` (λεξικογραφικά) Αρνητικό αριθμό αν `s1 < s2` (λεξικογραφικά)

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

int main()
{
    char c,s1[50],s2[50];
    gets(s2);
    cout << "You entered " << strlen(s2)
         << "  characters"<<endl;

    strcpy(s1,s2);
    cout <<"s1 = "<< s1<<endl<< "s2 = "<<s2<<endl;
    cout <<"compare s1 , s2 : " << strcmp(s1,s2)<<endl;

    strcat(s1,"A");
    cout << "s1 = "<<s1<<endl<< "s2 = "<<s2<< endl;
    cout <<"compare s1 , s2 : " << strcmp(s1,s2)<<endl;

    strcat(s2,"ABC");
    cout << "s1 = "<<s1 << endl<< "s2 = "<<s2<<endl;
    cout <<"compare s1 , s2 : " << strcmp(s1,s2)<<endl;

    return 0
}
```

Pointers - Δείκτες

```
int a;
```
Δέσμευση διεύθυνσης (περιοχής) στη μνήμη για ακέραιο.
address-of operator (&) :&a διεύθυνση της a (&a = 0x28fefc)

```
int *pa;
```
Δήλωση δείκτη σε ακέραιο. Δέσμευση διεύθυνσης ή οποία περιέχει τη διεύθυνση ακεραίου.
&pa = 0x28fef8 , pa = 0xfffffffe

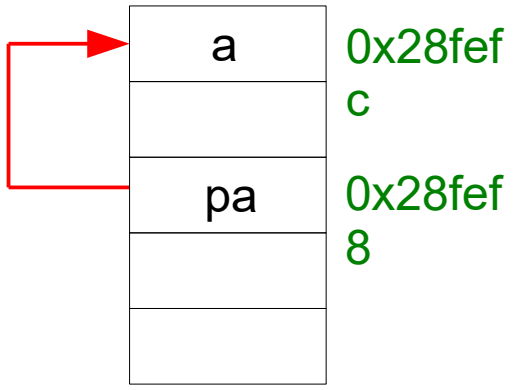
```
a =10;
```

```
pa = &a;
```
pa = 0x28fefc

```
int c
```

```
c = *pa;
```
περιεχόμενο της διεύθυνσης στην οποία δείχνει ο pa (c=a)
dereference operator (*)

```
pa = NULL;
```
pa = 0



Δείκτες και πίνακες

```
int i_array[10]={2,4,3,12,24,26,35,9,44,10};
int* i_p1,int* i_p2;
```

i_array διεύθυνση του πρώτου στοιχείου
i_array = &i_array[0]

```
a = *i_array;
```
a=2

```
i_p1 = i_array;
```

```
i_array = i_p2;
```
δεν επιτρέπεται. `i_array` σταθερός δείκτης

```
*(i_array+m)
```
*(i_array + m * n) n:αριθμός bytes

```
*(i_array + m) = i_array[m]
```

```
a = *(i_array++);
```
a=4

```
char* c_string="hello";
```

Δείκτες σε δείκτες

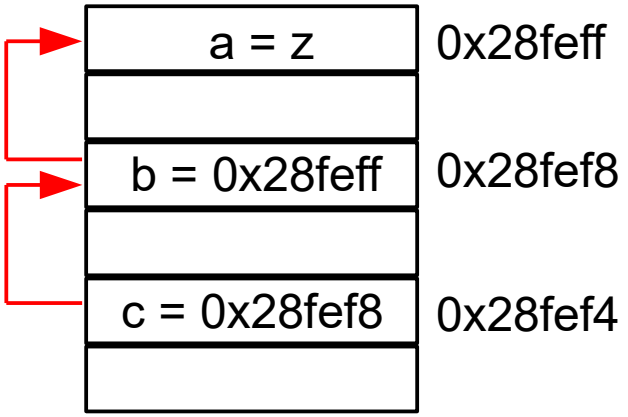
```
char a;
char * b;
char ** c;
a = 'z';
b = &a;
c = &b;
```

χαρακτηρας

δείκτης σε χαρακτήρα

δείκτης σε δείκτη χαρακτήρα

```
**c = z
```



Πίνακες δεικτών

```
int main()
{
   char *month[]={"JANUARY","FEBRUARY","MARCH","APRIL","MAY","JUNE",
        "JULY","AUGUST","SEPTEMBER","OCTOBER","NOVEMBER","DECEMBER"};
    for(int i=0; i<12; i++)
      cout << *month[i] << "  " << (int)month[i] << endl;
    for(int j=0; j<12 ;j++)
      cout << month[j] << endl;
   // cout << *(month+j);
}
```

J	46e065	JANUARY
F	46e06d	FEBRUARY
M	46e076	MARCH
A	46e07c	APRIL
M	46e082	MAY
J	46e086	JUNE
J	46e08b	JULY
A	46e090	AUGUST
S	46e097	SEPTEMBER
O	46e0a1	OCTOBER
N	46e0a9	NOVEMBER
D	46e0b2	DECEMBER

J	A	N	U	A	R	Y	\0		
F	E	B	R	U	A	R	Y	\0	
M	A	R	C	H	\0				
A	P	R	I	L	\0				
M	A	Y	\0						
J	U	N	E	\0					
J	U	L	Y	\0					
A	U	G	U	S	T	\0			
S	E	P	T	E	M	B	E	R	\0
O	C	T	O	B	E	R	\0		
N	O	V	E	M	B	E	R	\0	
D	E	C	E	M	B	E	R	\0	

Δείκτες void (void pointers)

`void *vp;` δείκτης σε μη προσδιορισμένο τύπο στοιχείου

```
int a;
double x[100];
char c;
```

Ο δείκτης `vp` μπορεί να δείχνει σε οποιοδήποτε τύπο στοιχείου

```
vp = &a;
vp = x;
vp = &c;
```

```
int b;
b = *vp;
```

δεν επιτρέπεται αφού δεν είναι γνωστός ο τύπος

```
int main()
{
  void *vp;
  int ia=55;
  float fx=5.5;;
  int iar[4]={10,20,30,40};

  vp=&ia;
  cout << *( (int*)vp ) << endl;          τυπώνει 55

  vp=&fx;
  cout << *( (float*)vp ) << endl;        τυπώνει 5.5

  vp=iar;
  cout << *( (int*)vp + 2 ) << endl;      τυπώνει 30
}
```

Δυναμική δέσμευση-αποδέσμευση μνήμης
New - new[]- delete

```
 //  C
double *dyn_x,*dyn_ar;
dyn_x = malloc(sizeof(double));
int n;
dyn_ar = malloc(n*sizeof(double));

free(dyn_x);
free(dyn_ar)
```

```
 //  C++
double *dyn_x,*dyn_ar;
dyn_x = new double;
int n;
dyn_ar = new double[n];


delete dyn_x;
delete[] dyn_ar;
```

- Εάν δεν είναι δυνατή η δέσμευση μνήμης παράγεται μια εξαίρεση (`bad_alloc exception`) και το πρόγραμμα τερματίζεται.
- Είναι δυνατόν να αλλάξουμε τη συμπεριφορά με τη παράμετρο `nothrow`.
- Δεν παράγεται εξαίρεση αλλά η new επιστρέφει δείκτη NULL (0);

```
int *pa, n ;
cin >> n;
pa new (nothrow) int[5];
if (pa == 0)
{
  cout << "Error allocating memory";
  ..................
  ..................
}
```

Παράδειγμα δυναμικής δέσμευσης μνήμης

```
int main()
{
  double *dyn_x,*dyn_ar; //δύο δείκτες σε double
                         // δεν δεσμεύεται μνήμη για τιμή, μόνο για δείκτες

  dyn_x = new double;    //δέσμευση μνήμης

  cout << "Enter coefficient :";
  cin >> *dyn_x;
  cout << endl;

  int n;
  cout << " Number of array elements :";
  cin >> n;
  cout << endl;
  dyn_ar = new double[n];//  δέσμευση μνήμης για τον πίνακα
```

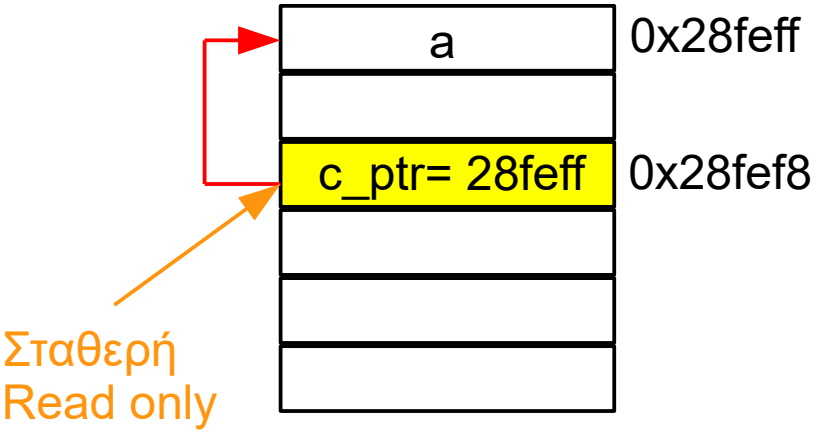
```
for(int i=0; i<n; i++)
  {
     cout << "array[" << i <<"] = ";
     cin >> dyn_ar[i];
     cout << endl;
  }
  for(int j=0; j<n; j++)
  {
     dyn_ar[j] *= *dyn_x;
     cout << dyn_ar[j];
     cout << endl;
  }
  delete dyn_x;     //  αποδέσμευση μνήμης
  delete[] dyn_ar; //  οι δείκτες δείχνουν στη μνήμη που αποδεσμεύτηκε

  dyn_x = NULL;
  dyn_ar = NULL;

  return 0;
}
```

Σταθεροί δείκτες

```
int a=1;
int * const c_ptr = &a;
```

- Η τιμή του δείκτη, η διεύθυνση στην οποία δείχνει, δεν μπορεί να αλλάξει
- Πρέπει να δοθεί τιμή στη δήλωση
- Η μεταβλητή στην οποία δείχνει δεν είναι σταθερή και μπορεί να αλλάξει

```
*c_ptr = 10;
```
a=10

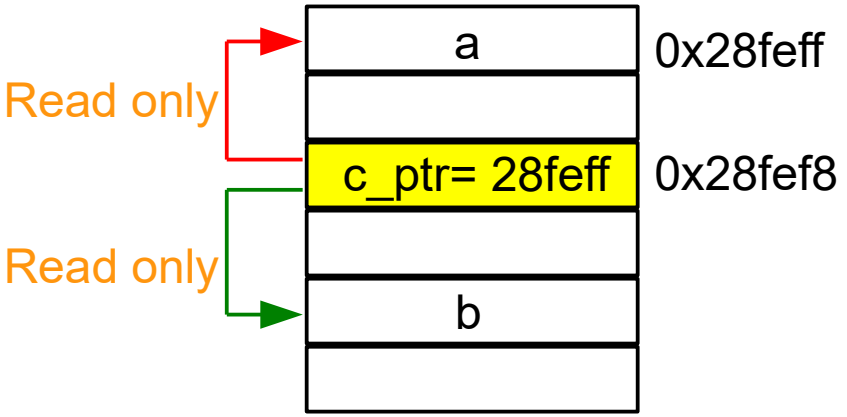



```
int a=1,b=2;
const int * c_ptr = &a;
```

- Η τιμή της a δεν είναι σταθερή και μπορεί να αλλάξει
- Η τιμή του δείκτη δεν είναι σταθερή και μπορεί να αλλάξει
- Η τιμή της a δεν μπορεί να αλλάξει μέσω του δείκτη

```
*c_ptr = 10;
```
δεν επιτρέπεται

```
c_ptr = &b;
```
επιτρέπεται




```
const int a;
const int * const c_ptr = &a
```
a σταθερή
σταθερός δείκτης

Μετατροπές τύπων σε Pointers

`reinterpret_cast` Μετατροπές μεταξύ ασύμβατων τύπων. Μετατρέπει ένα τύπο σε άλλο θεμελιωδώς διαφορετικό τύπο π.χ. ακέραιο σε pointer ή αντίστροφα.

```
int i;
char *p = "This is a string";
i = reinterpret_cast<int> (p);
```

`const_cast` Μετατρέπει ένα σταθερό δείκτη σε μεταβλητό δείκτη

```
int a = 50;
const int *pb=&a;
int *pv;
cout << " a = "<< a << endl;      a = 50
*pb = 100;    //Λάθος, σταθερός δείκτης
pv = const_cast<int*> (pb);
*pv = 100;
cout << " *pb = "<< *pb << endl;
cout << " a = "<< a << endl;      a = 100
```

Αναφορές References

```
int i,k;
i=10;

int & j = i;
*j =3;
j = 3;
j = k;
```

Αναφορά στο i,
όχι διεύθυνση του j
j συνώνυμο του i
λάθος, δεν είναι δείκτης
i =3
i=k

- Οι αναφορές είναι έμμεσες σταθερές (σταθεροί δείκτες) και πρέπει να αρχικοποιούνται
- &i = &j
- Η αναφορά δεν μπορεί να αλλάξει τιμή και να αναφέρεται σε άλλη μεταβλητή

- Χρήση σε συναρτήσεις

```
int i;
i=10;
const int & j = i;

i = 3;
j = 3;
```

επιτρέπεται
δεν επιτρέπεται

Δομές Structures

Array : όλα τα στοιχεία του ίδιου τύπου

Οριζόμενοι τύποι : στοιχεία διαφόρων τύπων

```
struct όνομα_δομής
{
   τύπος όνομα;
   τύπος όνομα;
   .
   .
}[ονόματα_μεταβλητών];
```

Μέλη δομής ή πεδία

```
όνομα_δομής όνομα_μεταβλητής;
```

struct

```
struct address
{
  char street[20];
  int number;
  char city[20];
  int code;
  char country[20];
};
int main()
{
  address work_address;
  address home_address;

  cout << sizeof(address);
  cin >> home_address.street;
  cin >> home_address.number;
  cin >> home_address.city;
  cin >> home_address.code;
  cin >> home_address.country;
  address work_address={"Panepistimiou",1000,
                        "Athens",12345,"Greece"};

  return 0;
}
```

Δήλωση δομής.
Δεν δεσμεύεται μνήμη

Δήλωση μεταβλητής .
Δεσμεύεται μνήμη 20+4+20+4+20=68bytes

"." τελεστής επιλογής μέλους

αρχικές τιμές

Δομή με μέλος άλλη δομή

```
struct address
{
  char street[20];
  int number;
  char city[20];
  int code;
  char country[20];
};
struct personal_info
{
  char first_name[25];
  char last_name[25];
  long home_phone;
  long work_phone;
  long mobile_phone;
  address home_address;
  address work_address;
};
int main()
{
  personal_info person_1;
  personal_info my_friends[10];
  cin >> person_1.work_address.street;

}
```

Δείκτες σε δομές

```
struct address
{
  char street[20];
  int number;
  char city[20];
  int code;
  char country[20];
};
int main()
{
  Address work_address={“Panepistimiou”,1000,
                        “Athens”,12345,”Greece”};
  address *p_address;

  p_address = &work_address;

  cout << p_address -> street; //(*p_address).street

  return 0;
}
```

*p_address.street *(p_address.street)

Unions

```
union u_type
{
   int i;
   char c;
   folat f;
   double d;
};
```

- όλα τα μέλη μοιράζονται την ίδια περιοχή μνήμης
- η ένωση δεσμεύει τόση μνήμη όση απαιτεί το μέλος με τα περισσότερα bytes
- Μόνο ένα μέλος πρέπει να χρησιμοποιείται κάθε φορά

```
union lenght_unit
{
  float meters;
  long milimeters;
}
struct book
{
  char title[50];
  char author[50];
  currency price;
};
```

Τύποι απαρίθμησης - Enumerated types

```
enum όνομα_τύπου
{
   τιμή1,τιμή2,τιμή3,..
} [ονόματα_μεταβλητών];
```

```
enum colors{black,blue,green,cyan,red,yellow,white};
```

Αυτόματη αρίθμηση: 0 1 2 3 4 5 6

```
enum colors{black=1,blue,green,cyan,red,yellow,white};
```

```
enum colors{black=2,blue,green=5,cyan,red,yellow,white};
```

3 6 7 8 9

typedef

Με την εντολή `typedef` δημιουργούμε συνώνυμα τύπων, όχι νέους τύπους

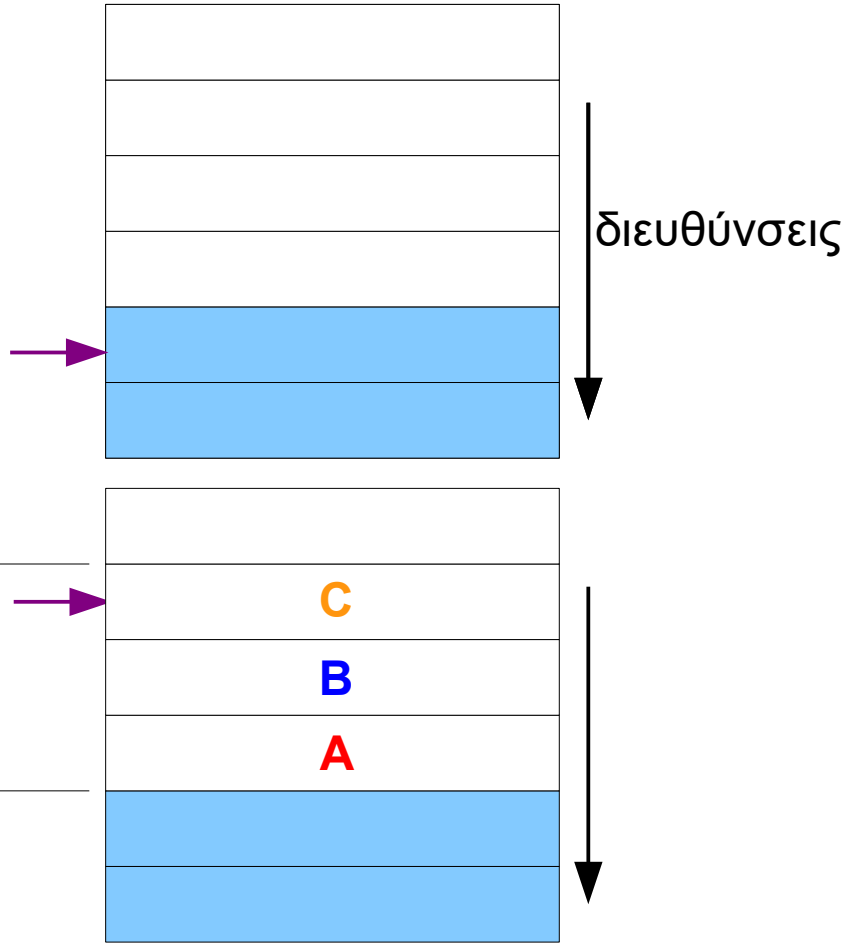
Συναρτήσεις

```
επιστρεφόμενος_τύπος όνομα_συνάρτησης(
                        τύπος_παραμέτρου παράμετρος_1,
                        τύπος_παραμέτρου παράμετρος_2,...)
{
   δηλώσεις σταθερών;
   δηλώσεις μεταβλητών; (τοπικές μεταβλητές)

   εντολές;

   return παράσταση;  (εκτός εάν ο επιστρεφόμενος_τύπος
                        είναι void)
}
```

Όπως και οι μεταβλητές η συνάρτηση πρέπει να οριστεί πριν από τη κλήση της

Στοίβα - stack

Περιοχή της μνήμης
με ιδιαίτερο τρόπο
προσπέλασης



αποθήκευση στη στοίβα

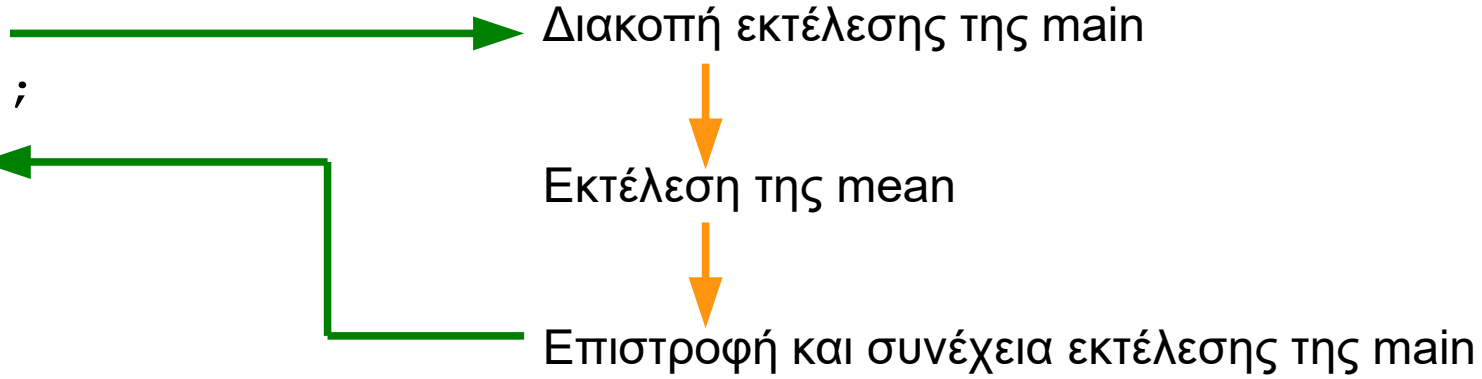
A
B
C

Ανάκληση από τη στοίβα

C
B
A

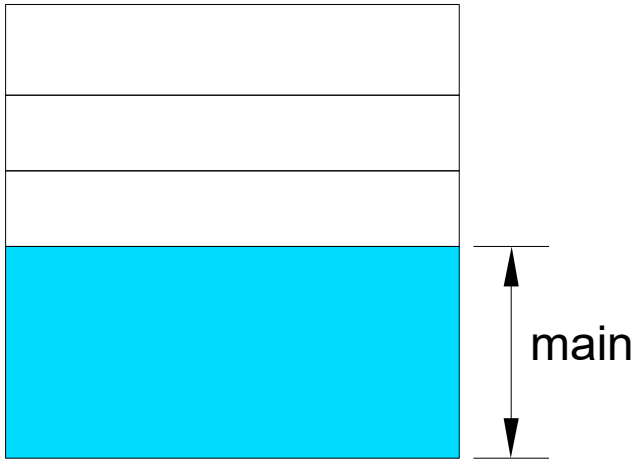
Last **I**n **F**irst **O**ut

```
double mean(double x, double y)
{
  double sum;
  sum = x + y;
  return sum/2;
}

int main()
(
  double x,y,z;
  cin >> x,y;

  z=mean(x,y);

  return 0;
}
```

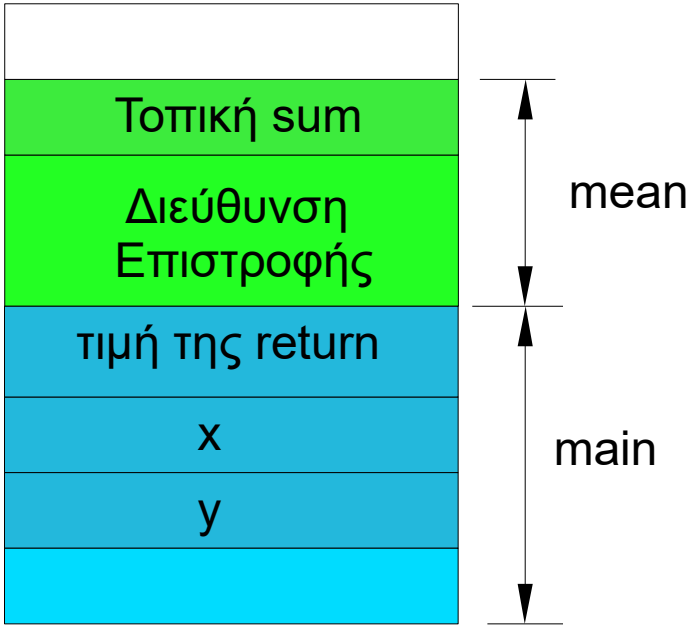


Stack frames

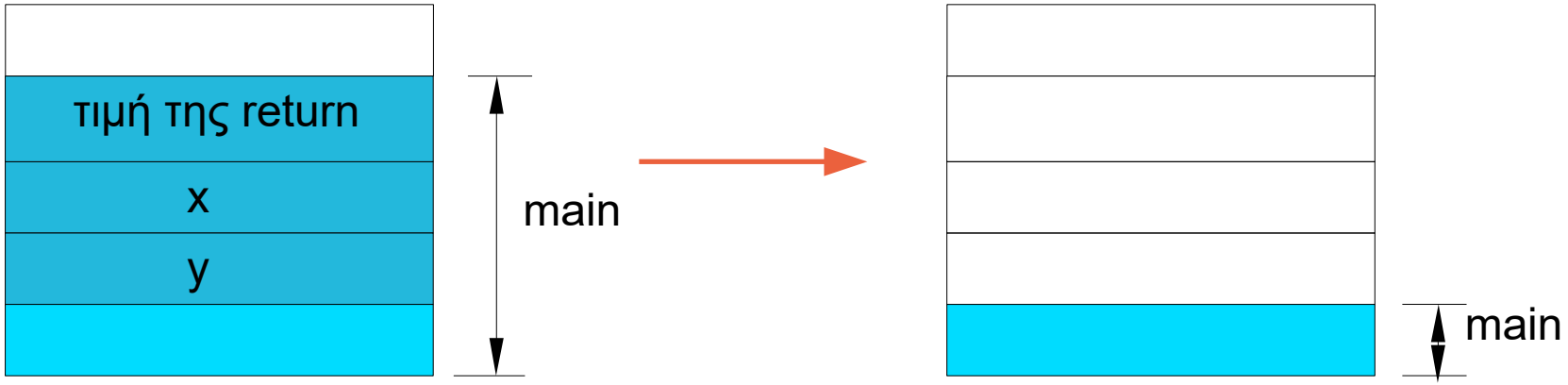
Πριν από τη κλήση της mean



Μετά τη κλήση της mean



Μετά το τέλος της mean

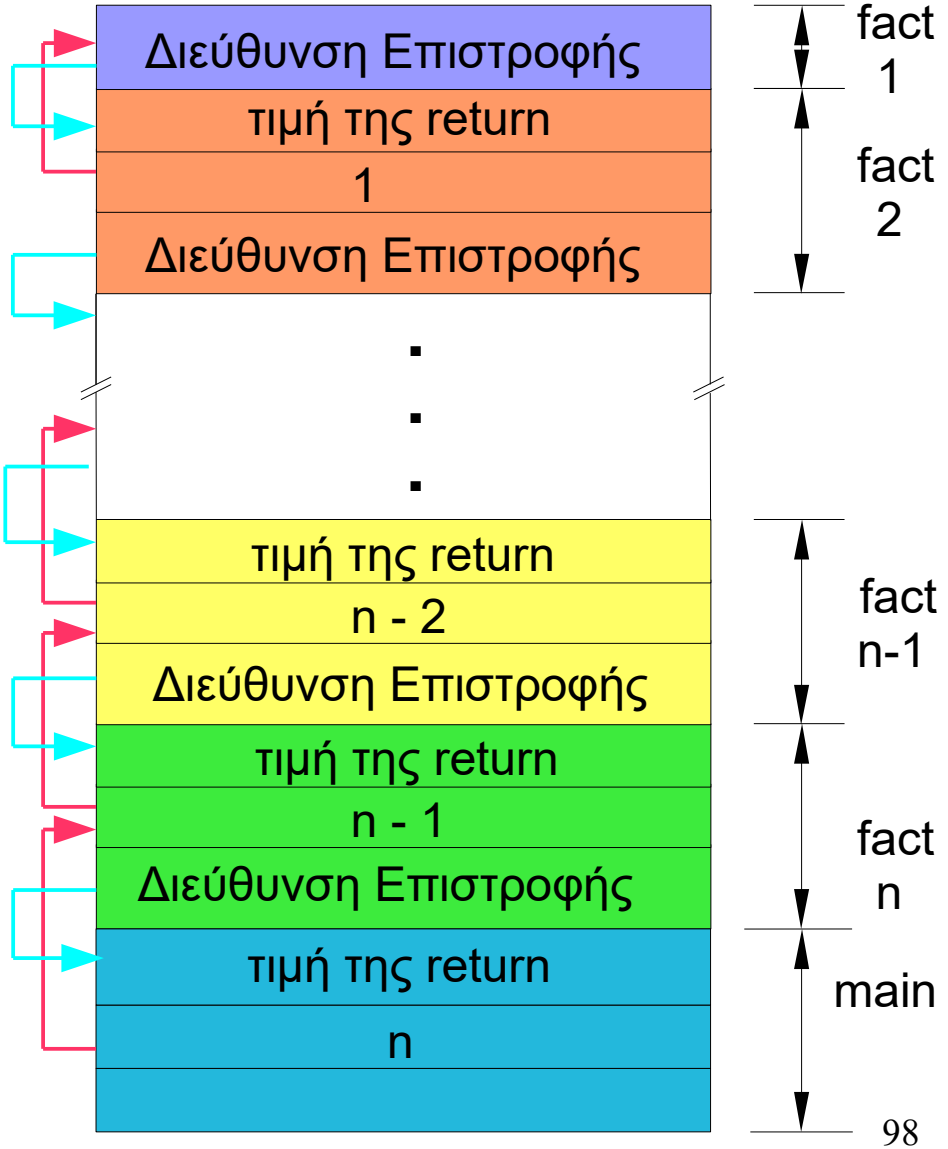


Αναδρομή Recursion

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

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

}
```

Έστω η κλήση `b = fact(-1);`
n < 0 : η συνθήκη τερματισμού της αναδρομής δεν μπορεί ισχύσει.

Stack overflow



Πέρασμα παραμέτρων με αντίγραφο τιμής

```
double mean(double , double );
```
δήλωση πρωτοτύπου
Forward declaration

```
int main()
{
  double x,y,z;
  cin >> x,y;
  z=mean(x,y);  //  mean(2,3) , mean(x*x/2,y*y/2)

  return 0;
}
double mean(double a, double b)
{
  double sum;
  sum = a + b;
  return sum/2;
}
```

παράμετροι με αντίγραφο τιμής
by value

- Παράμετροι : μεταβλητές, σταθερές, παραστάσεις.
- Η συνάρτηση δεν μπορεί να αλλάξει τις τιμές των παραμέτρων.
- Το πέρασμα μεγάλων δομών μπορεί να αυξήσει το χρόνο εκτέλεσης.
- Η συνάρτηση μπορεί να επιστέψει μόνο μία τιμή με τη `return`

Πέρασμα παραμέτρων με αναφορά

```
void swap_bv(int a, int b)
{
    int temp;
    temp = b;
    b = a;
    a = tem;
}
```

```
void swap_br(int& a, int& b)
{
    int temp;
    temp = b;
    b = a;
    a = temp;
}
```

```
int main()
{
    int x=20,y=10;
    swap_bv(x,y); //καμιά αλλαγή
    swap_br(x,y); //γίνεται ανταλλαγή
}
```

- Παράμετροι : μεταβλητές, όχι σταθερές, όχι παραστάσεις.
  - `swap_br(2,3)` λάθος
  - `swap_br(x/2,y/2)` λάθος
- Η κλήση με αναφορά είναι ίδια με τη κλήση με αντίγραφο τιμής

- Η συνάρτηση μπορεί να επιστρέφει περισσότερες από μία τιμές.
- Πέρασμα δομών χωρίς αντιγραφή όλων των στοιχείων.

Πέρασμα παραμέτρων με σταθερή αναφορά

```
#include <iostream>
#include <math.h>
using namespace std;
struct point
{
  double x,y;
}
double points_distance(const point& p1,const  point& p2)
{
  double dx,dy,dist;
  dx = p1.x - p2.x;
  dy = p1.y - p2.y;
  p1.x = 10.0;     λάθος
  return sqrt(dx*dx + dy*dy);
}
int main()
{
  point p1 = {2.0,5.0};
  point p2 = {10.0 30.0};
  cout << "Distance = " << points_distance(p1,p2) << endl;
  return 0;
}
```

Η συνάρτηση δεν μπορεί να αλλάξει τις τιμές των παραμέτρων

Πέρασμα παραμέτρων με διεύθυνση

```
void swap(int* , int*);
int main()
{
    int x=20,y=10;
    swap(&x,&y);
}
void swap(int* a, int* b)
{
    int temp;
    temp = *b;
    *b = *a;
    *a = tem;
}
```

Κλήση με παραμέτρους τις διευθύνσεις των μεταβλητών

Ορισμός της συνάρτησης :
Δείκτες στις μεταβλητές

- Παράμετροι : μεταβλητές, όχι σταθερές, όχι παραστάσεις.
  - `swap(&2,&3)` λάθος
  - `swap_br(&(x/2),&(y/2))` λάθος

- Η συνάρτηση μπορεί να επιστρέφει περισσότερες από μία τιμές.
- Πέρασμα δομών χωρίς αντιγραφή όλων των στοιχείων.

Συναρτήσεις με παραμέτρους πίνακες -1

```
int read_array (double data_points[], int max_elements)
//int read_array (double *data_points, int max_elements)
{
  int n;
  do
  {
    cout<<"Enter number of data points [2 to "<<max_elements<<"]:";
    cin >> n;
    cout << endl;
  }while ( (n < 2) || (n > 100) );

  for (int i=0 ; i < n ; i++)
  {
    cout << "x["<< i+1 << "] = ";
    cin >> data_points[i];
    cout << endl;
  }
  return n;
}
void print_array(double x[],int n)
{
   for(int i=0;i<n;i++)
       cout << "x[" << i << "] = "<< x[i] << endl;
}
```

Συναρτήσεις με παραμέτρους πίνακες -2

```
void calculations(double x[], int n,
                  double *max_value,double *min_value,
                  double *mean_value;double *standard_deviation)
{
  *min_value=x[0]
  *max_value=x[0];
  *mean_value=0.0;
  for(int j=0 ; j < n ;j++)
  {
    if (x[j] < *min_value) *min_value = x[j];
    if (x[j] > *max_value) *max_value = x[j];
    *mean_value += x[j];
  }
  *mean_value = *mean_value/n;

  *standard_deviation = 0.0;
  for(int j=0 ; j < n ; j++)
     *standard_deviation += pow(x[j]-*mean_value,2);
  *standard_deviation = sqrt(*standard_deviation/n);
}
```

Συναρτήσεις με παραμέτρους πίνακες - 3

```
void swap(double *xa, double *xb)
{
  double temp;
  temp = *xa;
  *xa = *xb;
  *xb = temp;
}
void comb_sort(double x[],int n)
{
   const float shrink = 1.3f;
   int offset;
   double temp;
   bool inOrder;
   offset = n;
   do
   {
     offset = (int)(offset/shrink);
     offset = (offset == 0) ? 1 : offset;
     inOrder = true;
     for(int i=0,j=offset;i<(n-offset);i++,j++)
     if (x[i] > x[j])
     {
       inOrder = false;
       swap(&x[i],&x[j]);
     }
   }while (!(offset == 1 && inOrder == true));
 }
```

```
int main()
{
  const int MAX = 100;
  double x[MAX];
  double max_value,min_value,mean_value,standard_deviation;
  int number_of_elements;

  number_of_elements = read_array(x,MAX);
  calculations(x, number_of_elements,&max_value,&min_value,
               &mean_value,&standard_deviation);
  cout << "Maximum value = " << max_value << endl;
  cout << "Minimum value = " << min_value << endl;
  cout << "Mean value    = " << mean_value << endl;
  cout << "Standard deviation = " << standard_deviation << endl;
  comb_sort(x,number_of_elements);
  cout << " Sorted array " << endl;
  print_array(x,number_of_elements);
  return 0;
}
```

Πρωτότυπα συναρτήσεων

```
int read_array (double*, int );
void calculations(double*,int, double*,double*,double*,double*);
void comb_sort(double*,int);
void print_array(double*,int);
```

```
int read_array (double data_points[], int );
void calculations(double x[], int n, double *max_value,
                  double *min_value, double *mean_value,
                  double *standard_deviation);
void comb_sort(double x[],int n);
void print_array(double x[] ,int n);
```

Συνάρτηση που επιστρέφει δείκτη Δυναμική δήλωση πίνακα

```
double* read_dynamic_array (int *n)
{
  do
  {
    cout << "Enter number of data points (minimum 2) ";
    cin >> *n;  cout << endl;
  }while ( *n < 2 );
  double* px = new double[*n]; //δημιουργία πίνακα
  if(px == NULL) return px; //δεν δημιουργήθηκε πίνακας
  for (int i=0 ; i < *n ; i++)
  {
     cout << "x["<< i+1 << "] = ";
     cin >> px[i];  // ή *(px+i)
     cout << endl;
  }
  return px; // επιστρέφει το δείκτη στον πίνακα
}

int main()
{
  int n;
  double *d_array;
  if( (d_array=read_dynamic_array(&n)) == NULL )
     cout << " Error ....";
```

Συνάρτηση που επιστρέφει δείκτη Δυναμική δήλωση δομής

```
struct address
{
  char street[20];
  int number;
  char city[20];
  int code;
  char country[20];
};

adress * alloc_address(int *n)
{
  address *p_address= new address[n];
  return p_address;
}

int main()
{
  int n;
  address *address = allo_address(&n);
  return 0;
}
```

Συναρτήσεις για πίνακες χαρακτήρων

```
int str_length(char str[])
{
  int i=0;
  while(str[i] !=0)
       i++;
  return i;
}
int str_length(char str[])
{
  int i=0;
  while(str[i++] !=0); // κενή εντολή
  return i-1;
}
int str_length(const char str[]) const:η συνάρτηση δεν μπορεί να
                                       αλλάξει τα στοιχεία του πίνακα
{
  int i=0;
  while(str[i++] !=0); // κενή εντολή
  return i-1;
}
int main()
{
  char str1[]="ABCDEFG";
  cout << "Length = " << str_length(str1);
}
```

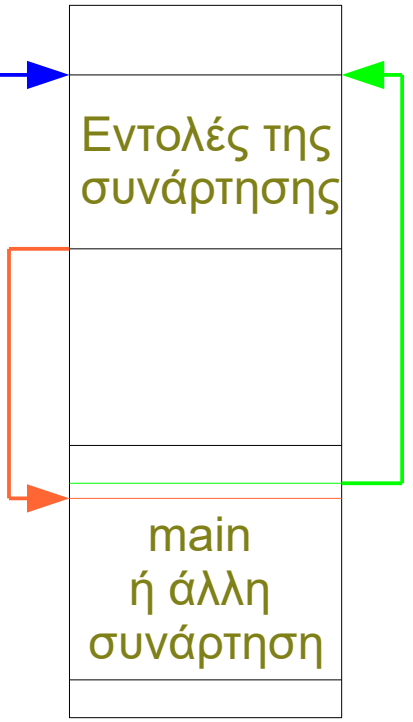
```
int str_length(const char *str)
{
  int i=0;
  while(*(str+i) !=0)
       i++;
  return i;
}

int str_length(const char *str)
{
  int i=0;
  while(*str++ !=0)
       i++;
  return i;
}
int str_length(const char *str)
{
  int i=0;
  while(*str++)
       i++;
  return i;
}
```

```
void copy_str(char *s1, const char *s2)
{
  int i = 0;
  while( (s1[i] = s2[i]) != 0 )
       i++;
}


void copy_str(char *s1,const char *s2)
{
  while( (*s1++ = *s2++) != 0 )
       ;
}

void copy_str(char *s1, const char *s2)
{
  while( *s1++ = *s2++ )
       ;
}
```

Δείκτες σε συναρτήσεις

```
int add_f(int a,int b)
{
   return a+b;
}
```

add_f διεύθυνση της συνάρτησης




```
int (*p_fun)(int , int);
int (*p_fun1)(int , int)=add_f;
```

δείκτης σε συνάρτηση
δείκτης σε συνάρτηση με αρχικοποίηση

```
int main()
{
 // int (*p_fun)(int , int);
  // int (*p_fun1)(int , int)=add_f;

   int a=2,b=3,c;

   p_fun = add_f;
   c = p_fun(a,b);
   cout << (*p_fun1)(5,3);
}
```

ισοδύναμες δηλώσεις

```
int (*p_fun)(int , int);
int *p_fun(int , int);
```

δείκτης σε συνάρτηση που επιστρέφει ακέραιο
η συνάρτηση επιστρέφει δείκτη σε ακέραιο

Συνάρτηση με όρισμα άλλη συνάρτηση

```
bool ascending(double x, double y)
{
    return y > x;
}

bool descending(double x, double y)
{
    return y < x;
}

bool (*comparison)(double, double);

void swap(double *xa, double *xb)
{
  double temp;
  temp = *xa;
  *xa = *xb;
  *xb = temp;
}
```

Συνάρτηση με όρισμα άλλη συνάρτηση

```
void comb_sort(double x[],int n,
               bool (*comparison)(double, double))
{
   const float shrink = 1.3f;
   int offset;
   double temp;
   bool inOrder;
   offset = n;
   do
   {
     offset = (int)(offset/shrink);
     offset = (offset == 0) ? 1 : offset;
     inOrder = true;
     for(int i=0,j=offset;i<(n-offset);i++,j++)
     if (comparison(x[i],x[j]))
     {
       inOrder = false;
       swap(&x[i],&x[j]);
     }
   }while (!(offset == 1 && inOrder == true));
 }
```

Κλήση της συνάρτησης ταξινόμησης

```
comb_sort(x,number_of_elements,descending);
```
για φθίνουσα

```
comb_sort(x,number_of_elements,ascending);
```
για αύξουσα

Πίνακες πολλών διαστάσεων ως ορίσματα

```
void matrix_mult(double ma[][MAXCOL], double mb[][MAXCOL],
          double mp[][MAXCOL],int row_a, int col_a,int col_b)
          //  pinakes m*n , n*p , m*p
{
   int i,j,k;
   for(i=0;i<row_a;++i)
     for(j=0;j<col_b;++j)
     {
        mp[i][j] = 0.0;
        for(k=0;k<col_a;++k)
           mp[i][j] += ma[i][k]*mb[k][j];
     }
}
```

Για όλες τις διαστάσεις εκτός από τη πρώτη πρέπει να δίνονται τιμές

Δυναμική δήλωση πίνακα δύο διαστάσεων

```
const int MAXCOL = 10;
const int MAXROW = 10;


double ** matrix_mult_2(double ma[][MAXCOL], double mb[][MAXCOL],
                        int row_a, int col_a,int row_b, int col_b)
{                        /* pinakes m*n , n*p , m*p */
    int i,j,k;
     double **mp;
     mp = new double*[col_b];          δημιουργία πίνακα δεικτών
     if (col_a==row_b)
         for(int i=0;i<col_a;i++)
            mp[i]=new double[row_a]; δημιουργία γραμμών του πίνακα
         else                        δύο διαστάσεων
            return NULL;

    for(i=0;i<row_a;++i)
       for(j=0;j<col_b;++j)
       {  mp[i][j] = 0.0;
          for(k=0;k<col_a;++k)
             mp[i][j] += ma[i][k]*mb[k][j];
       }
    return mp;
}
```

Δυναμική δήλωση πίνακα δύο διαστάσεων

```
int main()
{
   double m1[MAXROW][MAXCOL]={{1,2,3},{4,5,6}};
   double m2[MAXROW][MAXCOL]={{1,2},{3,4},{5,6}};

    int row_1=2,col_1=3,row_2=3,col_2=2;

   double **mp2;

   mp2 = matrix_mult_2(m1,m2,row_1,col_1,row_2,col_2);

   for(int j=0; j< row_1; j++)
   {
      delete[] mp2[j];
      mp2[j] = NULL;
   }

}
```

```
const int row=4;
const int col=4;
double m1[row][col]={{11,12,13,14},
                     {21,22,23,24},
                     {31,32,33,34},
                     {41,42,43,44}};



for(int i=0,j=0; i< row; i++,j++)
{
   cout << m1[i][j];
   cout endl;
}
```

inline

- Ο μηχανισμός κλήσης μιας συνάρτησης (διακοπή προγράμματος – επιστροφή) προκαλεί καθυστέρηση στην εκτέλεση.
- Με την εντολή `inline` αντί για τη κλήση ενσωματώνεται ο κώδικας της συνάρτησης στο πρόγραμμα.

```
inline void swap(double *xa, double *xb)
{
  double temp;
  temp = *xa;
  *xa = *xb;
  *xb = temp;
}
```

Υπερφόρτωση συναρτήσεων
Function overloading

Δύο ή περισσότερες συναρτήσεις με το ίδιο όνομα μπορούν να διαφέρουν:

- Στο τύπο των παραμέτρων
- Στο πλήθος των παραμέτρων
- Στον τύπο της τιμής που επιστρέφουν αλλά μόνο αν διαφέρουν και σε μία τουλάχιστον παράμετρο

Ο μεταγλωττιστής επιλέγει ποια από τις συνώνυμες συναρτήσεις θα χρησιμοποιήσει ανάλογα με τρόπο κλήσης (τύπος παραμέτρων).

Υπερφόρτωση της συνάρτησης swap

Παράμετροι με διεύθυνση (δείκτη)

```
void swap(int *a, int *b)
{
  int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}
void swap(double *a, double *b)
{
  double temp;
  temp = *a;
  *a = *b;
  *b = temp;
}
void swap(char *a, char *b)
{
  char temp;
  temp = *a;
  *a = *b;
  *b = temp;
}
```

Παράμετροι με αναφορά

```
void swap(int &a, int &b)
{
  int temp;
  temp = a;
  a = b;
  b = temp;
}
void swap(double &a, double &b)
{
  double temp;
  temp = a;
  a = b;
  b = temp;
}
void swap(char &a, char &b)
{
  char temp;
  temp = a;
  a = b;
  b = temp;
}
```

Παράμετροι με διεύθυνση (δείκτη)

```
int main()
{
  double x=1.2, y=3.5;
  int a=5 , b = 10;
  char c1 = 'A' , c2 = 'M';

  swap(&x,&y);
  swap(&a,&b);
  swap(&c1,&c2);
}
```

Παράμετροι με αναφορά

```
int main()
{
  double x=1.2, y=3.5;
  int a=5 , b = 10;
  char c1 = 'A' , c2 = 'M';

  swap(x,y);
  swap(a,b);
  swap(c1,c2);
}
```

Υπερφορτωμένη συνάρτηση `read_array`

```
const int max_rows=10;
const int max_columns=10;

void read_array(double array1d[],int n)
{
    for(int i=0;i<n;i++)
    {
        cout << "x["<<i<<"] = ";
        cin >> array1d[i]; cout << endl;
    }
}

void read_array(double array2d[][max_columns],int row, int col)
{
    for(int r=0;r<row;r++)
    {
        for(int c=0;c<col;c++)
        {
            cout << "x["<<r <<" , " << c <<"] = ";
            cin >> array2d[r][c]; cout << endl;
        }
    }
}
```

Υπερφορτωμένη συνάρτηση `read_array`

```
int main()
{
  const int max_rows = 20;
  const int max_columns = 20;
  int rows=3,columns=4;
  double array1d[max_columns];
  double array2d[max_rows][max_columns];

  read_array(array1d,columns);
  read_array(array2d,rows,columns);


}
```

Προκαθορισμένες τιμές παραμέτρων
Default parameters

Είναι δυνατό να δώσουμε αρχικές τιμές των παραμέτρων στον ορισμό μιας συνάρτησης. Οι τιμές αυτές θα χρησιμοποιηθούν αν δεν δώσουμε τιμές κατά τη κλήση.

```
int fun(int a=1, int b=2, int c=3)
{  }

fun();
fun(10,20,30);
fun(10);
```

Δεν επιτρέπεται να δώσουμε αρχική τιμή στην a ή στην b αλλά όχι στην c.

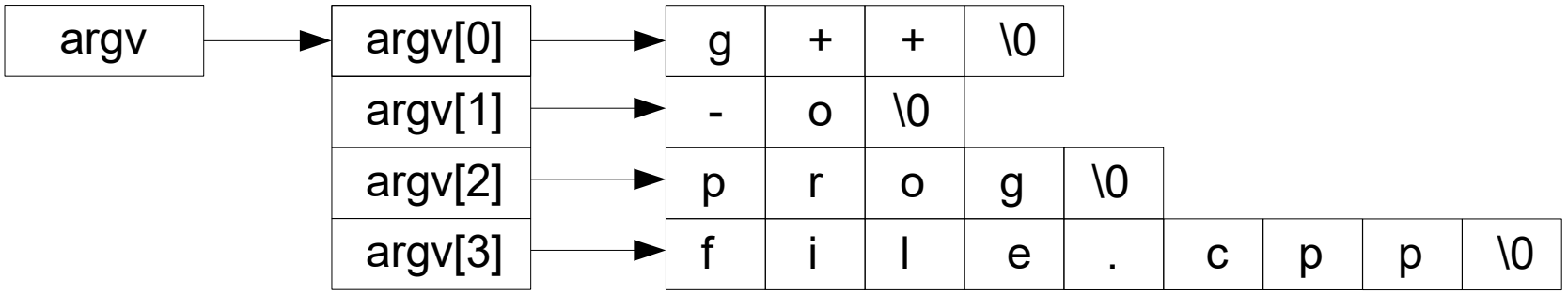
```
int fun(int a=1, int b=2, int c)    λάθος
int fun(int a  , int b=2, int c)    λάθος
int fun(int a=1, int b  , int c=3)  λάθος

int fun(int a, int b  , int c=3)    σωστό
int fun(int a, int b=2, int c=3)    σωστό
```

```
#include <iostream>
#include <math.h>
using namespace std;
double points_distance(double x1, double y1,
                       double x2=0.0, double y2=0.0)
{
  return sqrt(pow(x1-x2,2) + pow(y1-y2,2));
}
int main()
{
  double x1,x2,y1,y2;

  cout << "Distance between points ";
  cout << points_distance(x1,x2,y1,y2)<<endl;
  cout << "Distance from origin ";
  cout << points_distance(x1,x2)<<endl;

  return 0;
}
```

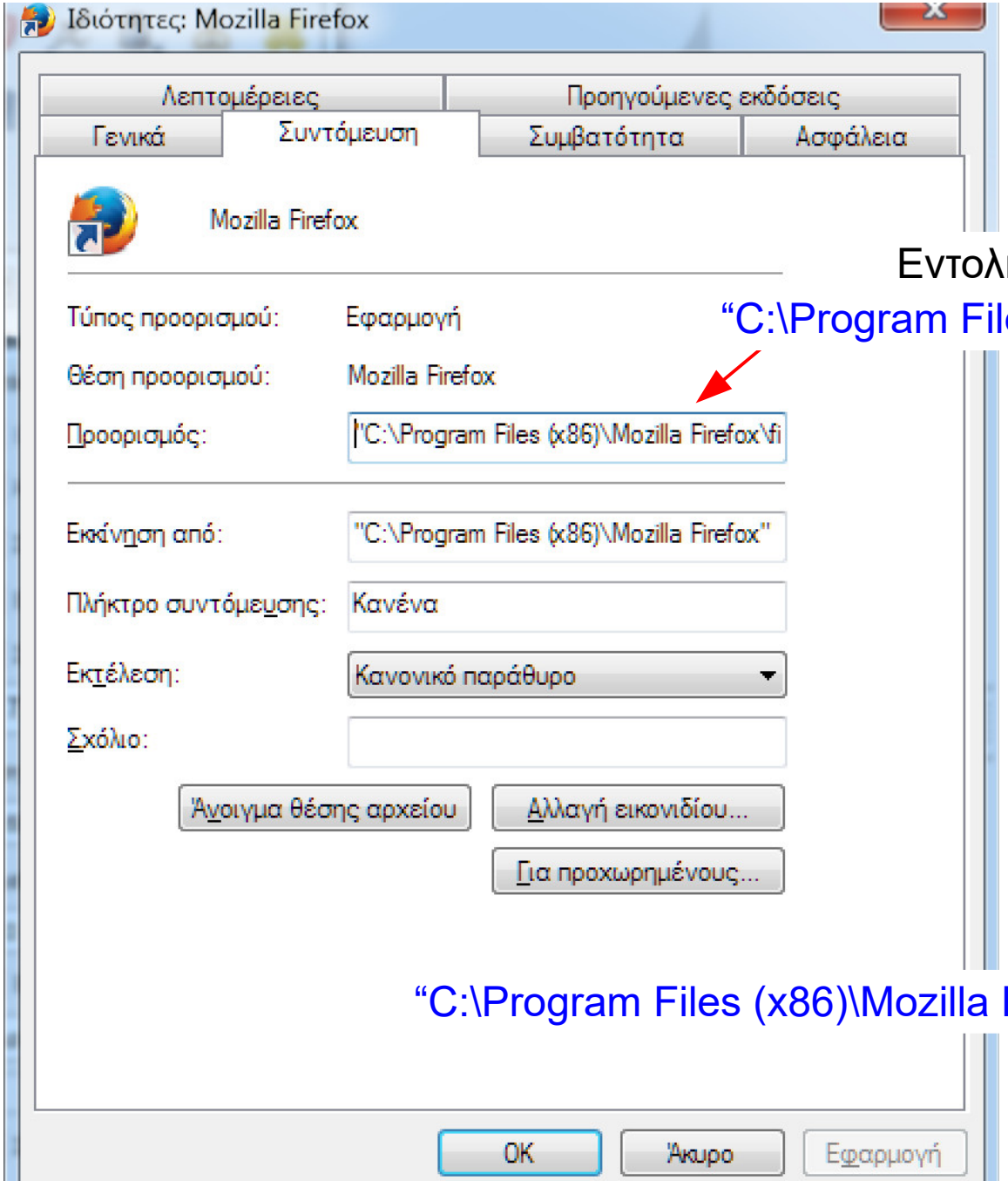
Παράμετροι γραμμής εντολών
Command line arguments

```
int main(int argc, char *argv[])
int main(int argc, char **argv)
```

g++ -o prog file.cpp
argc = 4 (0 , 1 , 2 , 3)



```
int main(int argc, char *argv[])
{
  if argc < n
  {
    cout << "Usage progname par1 par2 par3"
    return 1;
  }
  ............
  ............
```



Εντολή εκκίνησης του προγράμματος

"C:\Program Files (x86)\Mozilla Firefox\firefox.exe"

"C:\Program Files (x86)\Mozilla Firefox\firefox.exe" *URL ή όνομα αρχείου*

Στατικές μεταβλητές - Δήλωση static

- Μια global μεταβλητή με το χαρακτηρισμό static ισχύει μόνο στο αρχείο που είναι δηλωμένη.
- Δεν μπορεί να γίνει ορατή σε άλλο αρχείο όπως γίνεται με τη δήλωση extern

- Με τη δήλωση πρέπει να δοθεί και αρχική τιμή
- Η αρχικοποίηση γίνεται μόνο τη πρώτη φορά
- Μία στατική μεταβλητή διατηρεί τη τιμή της έως το τέλος του προγράμματος

```
int count_calls()
{
  static int counter = 0;
  return ++counter;
}

int main()
{
   for (int i = 0; i != 10;i++)
      cout << count_calls()<<endl;
   return 0;
}
```

```
#include <iostream>
using namespace std;
int r_avg(int i);

int main()
{
  int num;
  do
  {
    cout << "Enter numbers (-1 to quit): ";
    cin >> num;
    if(num != -1)
      cout << "Running average is: " << r_avg(num);
    cout << endl;
} while(num > -1);
return 0;
int r_avg(int i)
{
   static int sum=0, count=0;
   sum = sum + i;
   count++;
   return sum / count;
}
```

Μεταβλητές register

- Με τη δήλωση `regiter` ζητάμε από τον μεταγλωττιστή να αποθηκεύσει τη τιμή της μεταβλητής όχι στη μνήμη αλλά σε ένα καταχωρητή της CPU.
  - `regsiter int a`
- Αν δεν υπάρχει διαθέσιμος καταχωρητής ο μεταγλωττιστής αγνοεί τη δήλωση `regiter`.
- Μια μεταβλητή μπορεί να δηλωθεί ως `regiter` όταν χρησιμοποιείται συχνά, π.χ. σε μία εντολή for ή while με μεγάλο πλήθος επαναλήψεων.

- Ο τύπος `regiter` χρησιμοποιήθηκε για πρώτη φορά στην C γιατί εξασφαλίζει τη ταχύτερη δυνατή πρόσβαση στη μεταβλητή κα αφορούσε μεταβλητές `int` και `char`

```
// Χρόνος εκτέλεσης με μεταβλητή register

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

int main()
{
  unsigned int i;
  register unsigned int j;

  unsigned int delay;
  long start_time, end_time;

  start_time = clock();
  for(delay=0; delay<50; delay++)
    for(i=0; i < 64000000; i++);
  end_time = clock();
  cout << "Number of clock ticks for non-register loop: ";
  cout << end_time-start_time << '\n';

  start_time = clock();
  for(delay=0; delay<50; delay++)
    for(j=0; j < 64000000; j++) ;
  end_time = clock();
  cout << "Number of clock ticks for register loop: ";
  cout << end_time-start_time << '\n';
  return 0;
```

Χώροι ονομάτων - namespaces

- Οι χώροι ονομάτων χρησιμοποιούνται στη C++ για να αποτρέπουν τη δημιουργία “διενέξεων” όταν υπάρχουν τα ίδια ονόματα μεταβλητών, συναρτήσεων ή κλάσεων σε περισσότερες από μια βιβλιοθήκες.
- Η C++ έχει ένα χώρο ονομάτων για τις βασικές βιβλιοθήκες: `std` (standard namespace). Παραδείγματα είναι οι `cin , cout.`
- Βασίζονται στον τελεστή εμβέλειας “`::`” .
- Εάν σε μία βιβλιοθήκη `mylib` ορίζεται μια άλλη συνάρτηση με το όνομα `cout` η διάκριση γίνεται με το τρόπο κλήσης `mylib::cout – std::cout`
- `using.` Οδηγία (directive) προς τον μεταγλωττιστή να συμπεριλάβει στο πρόγραμμα βιβλιοθήκες που έχουν διαφορετικούς χώρους ονομάτων.
- Η χρήση των χώρων ονομάτων εγγυάται ότι σε μια περιοχή του προγράμματος (π.χ. block, συνάρτηση) τα ονόματα είναι μοναδικά.

Παράδειγμα namespace

```
#include <iostream>
using namespace std;
namespace first
{
  int x = 5;
}
namespace second
{
  double x = 3.1416;
}
int main ()
{
  using namespace first;
  {
    using namespace first;
    cout << x << endl;
  }
  {
    using namespace second;
    cout << x << endl;
  }
  return 0;
}
```

Καθολική Εμβέλεια χώρου ονομάτων `std`

λάθος. Εμβέλεια όλη η `main`

Εμβέλεια χώρου ονομάτων `first`

Εμβέλεια 3 χώρου ονομάτων `second`

error: reference to 'x' is ambiguous

- Παραδοσιακός προγραμματισμός
  - Δεδομένα
  - Συναρτήσεις (εντολές) επεξεργασίας των δεδομένων

- Αντικείμενα
  - Ιδιότητες (σχήμα, χρώμα, μάζα, ...)
  - Ενέργειες που μπορούν να κάνουν ή να γίνουν σε αυτά

```
struct Box
{
   double length;
   double width;
   double height;
};
```

```
#include <iostream>
using namespace.std
int main( )
{
   Box Box1;
   double volume = 0.0;

   Box1.height = 5.0;  // cin >> Box1.height;
   Box1.length = 6.0;  // cin >> Box1.length;
   Box1.width  = 7.0;  // cin >> Box1.width;

   volume = Box1.height * Box1.length * Box1.width;
   cout << "Volume of Box1 : " << volume <<endl;
   return 0;
}
```

Object-Oriented Programming

Ορίζουμε αντικείμενα που έχουν

- Χαρακτηριστικά
  - Ιδιότητες (properties)
  - Πεδία (fields)
  - Γνωρίσματα (attributes)
- Συμπεριφορά
  - Μεθόδους

Κλάσεις Classes

```
class <όνομα κλάσης>
{
  private:
      <δηλώσεις>     //ιδιωτικό μέρος
  public:
      <δηλώσεις>     //δημόσιο μέρος
};  εδώ πρέπει να υπάρχει ο χαρακτήρας ";"
```

```
struct <όνομα δομής>
{
    <δηλώσεις>
};
```

Κλάσεις - Δομές

- Στη C++ οι δομές μπορούν να περιέχουν τόσο δεδομένα όσο και συναρτήσεις
- Οι λέξεις `class` και `struct` μπορούν να χρησιμοποιηθούν
- Η μόνη διαφορά είναι ότι τα μέλη της δομής είναι δημόσια ενώ η κλάση μπορεί να έχει και ιδιωτικά μέλη
- Οι ενώσεις (unions) διαφέρουν από τις δομές στο ότι μπορούν να έχουν ένα μέλος δεδομένων σε κάθε στιγμή. Μπορούν όμως να έχουν συναρτήσεις μέλη. Οπως και με τις δομές η πρόσβαση στα μέλη των ενώσεων είναι δημόσια.

Κλάσεις (Classes)

```
class Box
{
 public:
  double m_length;
  double m_width;
  double m_height;

  void BoxDimentions(double length,double width,double height)
  {
     m_length = length;
     m_width  = width;
     m_height = height;
  }

  double BoxVolume()
  {
     return m_length * m_width * m_height ;
  }

};
```

Member variables

Member functions

Στιγμιότυπο (instance)

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

int main( )
{
    Box Box1;   //δήλωση μεταβλητής-αντικειμένου (instantiating class)
    Box1.BoxDimentions(5.0 , 6.0 , 7.0);
    cout << "Volume of Box1 : " << Box1.BoxVolume() << endl;
    return 0;
}
```

Προσδιοριστικά πρόσβασης
access specifiers

Δικαίωμα πρόσβασης σε μεταβλητές ή συναρτήσεις με χαρακτηρισμό :

- **private** (ή χωρίς). Μόνο μέσα από τη κλάση
- **public.** Από τη main ή άλλες συναρτήσεις
- **protected.** Ειδικά δικαιώματα πρόσβασης για κληρονομικότητα inheritance
  - Η πρόσβαση είναι δυνατή μόνο από όπου είναι “ορατή” η κλάση

```
class Box
{
 private:
  double m_length;
  double m_width;
  double m_height;

 public:
  void BoxDimentions(double length,double width,double height)
  {
     m_length = length;
     m_width = width;
     m_height = height;
  }

  double BoxVolume()
  {
     return m_length * m_width * m_height ;
  }

};
```

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

int main( )
{
    Box Box1;   //δήλωση μεταβλητής-αντικειμένου (instantiating class)

    Box1.BoxDimentions(5.0 , 6.0 , 7.0);

    Box1.m_height = 10.0; //  σφάλμα στη μεταγλώττιση

    cout << "Volume of Box1 : " << Box1.BoxVolume() << endl;
    return 0;
}
```

Εγκλωβισμός (encapsulation)

Η πρόσβαση στα μέλη μιας κλάσης επιτρέπεται μόνο μέσω μιας συνάρτησης (μεθόδου) – Απόκρυψη πληροφορίας (information hiding)

- Οι τιμές των μελών δεν μπορούν να αλλάξουν από οπουδήποτε
- Είναι δυνατόν να ελέγχεται η ορθότητα των τιμών

```
class Box
{
 private:
  double m_length;
  double m_width;
  double m_height;

 public:
  int BoxDimentions(double length,double width,double height)
  {
    // η συνάρτηση επιστρέφει ένα κωδικό σφάλματος
    if ((length >0.0) && (width >0.0) && (height >0.0))
      {
        m_length = length;
        m_width = width;
        m_height = height;
        return 0;
      }
    else
       return -1;
  }

  double BoxVolume()
  {
     return m_length * m_width * m_height ;
  }

};
```

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

int main( )
{
    Box Box1;
    int error_code;

    error_code=Box1.BoxDimentions(5.0 , 6.0 , 7.0);
    if (error_code == 0)
        cout << "Volume of Box1 : " << Box1.BoxVolume() << endl;
    else
        cout <<"Error "<<error_code<<"  Λάθος διαστάσεις"<<endl;
    return 0;
}
```

Αρχικοποίηση αντικειμένων

```
class Date
{
 private:
    int m_nMonth;
    int m_nDay;
    int m_nYear;
};
 int main()
{
    Date cDate;
    // cDate's member variables now contain garbage
    // Who knows what date we'll get?

    return 0;
}
```

Συναρτήσεις κατασκευής (constructors)

- Συνάρτηση (ή συναρτήσεις) που δίνει αρχικές τιμές στα μέλη μίας κλάσης όταν δηλώνουμε ένα αντικείμενο (αρχικοποίηση)
- Η συνάρτηση έχει το ίδιο όνομα με τη κλάση και δεν έχει τύπο επιστρεφόμενης τιμής, ούτε void.
- Η συνάρτηση κατασκευής χωρίς ορίσματα ονομάζεται default constructor.

```
class Box
{
 private:
  double m_length;
  double m_width;
  double m_height;

 Public:
  Box()      //default constructor
  {
    m_length = 1.0;
    m_width  = 1.0;
    m_height = 1.0;
  }

  void BoxDimentions(double length,double width,double height)
  {
     m_length = length;
     m_width = width;
     m_height = height;
  }

  double BoxVolume()
  {
     return m_length * m_width * m_height ;
  }
};
```

```
class Box
{
 private:
  double m_length,m_width,m_height;
 public:

  Box()
  { //default constructor με αρχικές τιμές.
    m_length = 1.0;
    m_width  = 1.0;
    m_height = 1.0;
  }
  Box(double length,double width,double height)
  { //constructor με παραμέτρους
    m_length = length;
    m_width  = width;
    m_height = height;
  }
  void BoxDimentions(double length,double width,double height)
  {
     m_length = length;
     m_width = width;
     m_height = height;
  }
  double BoxVolume()
  { return m_length * m_width * m_height ;}
};
```

Λίστα αρχικοποίησης

:όνομα_μεταβλητής(τιμή),όνομα_μεταβλητής(τιμή),....

Χωρίς ; στο τέλος

```
class Box
{
 private:
  double m_length;
  double m_width;
  double m_height;

public:
  Box():m_length(1.0),m_width(1.0),m_height(1.0) //default constructor
  {}
  Box(double length,double width,doubleheight)
       :m_length(length),m_width(width),m_height(height)
  {}
  void BoxDimentions(double length,double width,double height)
  {
     m_length = length;
     m_width = width;
     m_height = height;
  }
  double BoxVolume()
  { return m_length * m_width * m_height ;}
};
```

```
class Box
{
 private:
  double m_length,m_width,m_height;
 public:

  Box(double length=1.0,double width=1.0,double height=1.0)
  { //constructor με παραμέτρους και default τιμές
    m_length = length;
    m_width  = width;
    m_height = height;
  }
  void BoxDimentions(double length,double width,double height)
  {
     m_length = length;
     m_width = width;
     m_height = height;
  }
  double BoxVolume()
  {
     return m_length * m_width * m_height ;
  }
};
```

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

int main( )
{
   Box Box1;                 //αρχικές τιμές (1.0 , 1.0 , 1.0)
// Box Box1(2.0,3.0,4.0); //αρχικές τιμές (2.0 , 3.0 , 4.0)
// Box Box1(2.5);          //αρχικές τιμές (2.5 , 1.0 , 1.0)
   int error_code;

    error_code=Box1.BoxDimentions(5.0 , 6.0 , 7.0);
   if (error_code == 0)
       cout << "Volume of Box1 : " << Box1.BoxVolume() << endl;
   else
        cout <<"Error "<<error_code<<"  Λάθος διαστάσεις"<<endl;
    return 0;
}
```

Τελεστής εμβέλειας (scope operator) (::)

```
class Box
{
 private:
  double m_length,m_width,m_height;
 public:

  Box(double length=1.0,double width=1.0,double height=1.0) ;
  void BoxDimentions(double length,double width,double height);
  double BoxVolume();
};

Box::Box(double length=1.0,double width=1.0,double height=1.0)
{ //constructor με παραμέτρους και default τιμές
    m_length = length;
    m_width  = width;
    m_height = height;
}
void Box::BoxDimentions(double length,double width,double height)
{
    m_length = length;
    m_width = width;
    m_height = height;
}
double Box::BoxVolume()
{
    return m_length * m_width * m_height ;
}
```

Η συνάρτηση κατασκευής καλεί άλλη συνάρτηση

```
class Box
{
 private:
  double m_length,m_width,m_height;
 public:

  Box(double length=1.0,double width=1.0,double height=1.0)
  {
     BoxDimentions(length,width,height)
  }
  void BoxDimentions(double length,double width,double height)
  {
     m_length = length;
     m_width = width;
     m_height = height;
  }
  double BoxVolume()
  {
     return m_length * m_width * m_height ;
  }
};
```

Ιδιωτικές συναρτησεις κατασκευής
Private constructors

```
class Box
{
 private:
  double m_length,m_width,m_height;
  Box()  //default constructor
  {  m_length = m_width  = m_height = 1.0;  }
 public:
  Box(double length,double width,double height)
  { //constructor με παραμέτρους
    m_length = length;
    m_width  = width;
    m_height = height;
  }
  void BoxDimentions(double length,double width,double height)
  {
     m_length = length;
     m_width = width;
     m_height = height;
  }
  double BoxVolume()
  { return m_length * m_width * m_height ; }
};
```

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

int main( )
{
  Box Box1;    //αυτή η δήλωση δεν επιτρέπεται γιατί η προκαθορισμένη
               // συνάρτηση (default constructor)  είναι ιδιωτική

  Box Box1(2.0,3.0,4.0); //αυτή η δήλωση  επιτρέπεται

   int error_code;

  error_code=Box1.BoxDimentions(5.0 , 6.0 , 7.0);
  if (error_code == 0)
       cout << "Volume of Box1 : " << Box1.BoxVolume() << endl;
  else
       cout <<"Error "<<error_code<<"  Λάθος διαστάσεις"<<endl;
  return 0;
}
```

implicit default constructor

Αν δεν δηλώσουμε constructor ο μεταγλωττιστής δημιουργεί ένα χωρίς παραμέτρους.

```
class Box
{
   double m_lemgth,m_width,m_height;
 public:
   void BoxDimentions(double length,
                      double width,
                      double height);
   double BoxVolume();
};
```

Η δήλωση αντικειμένων δεν πρέπει να έχει παραμέτρους

```
main
{
  Box Box1;
 ........
}
```

Copy constructors

Επιτρέπει τη δημιουργία αντικειμένων με αντιγραφή των δεδομένων άλλου αντικειμένου της ίδιας κλάσης που υπάρχει

```
class Box
{
 private:
  double m_length,m_width,m_height;
  Box() //default constructor
  { m_length = m_width  =  m_height = 1.0; }
 public:
  Box(double length,double width,double height)//constructor με παραμέτρους
  { m_length = length; m_width  = width; m_height = height; }

  Box (const Box &BoxCopy) //copy constructor
  {  m_length = BoxCopy.m_length;
     m_width = BoxCopy.m_width;
     m_height = BoxCopy.m_height;
  }

  void BoxDimentions(double length,double width,double height)
  { m_length = length; m_width = width; m_height = height; }

  double BoxVolume()
  { return m_length * m_width * m_height ; }
};
```

Copy constructors

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

int main( )
{
  Box Box1(2.0,3.0,4.0);
  Box Box2(Box1);

  cout << "Volume of Box1 : " << Box1.BoxVolume() << endl;
  cout << "Volume of Box2 : " << Box2.BoxVolume() << endl;
  return 0;
}
```

Συνάρτηση καταστροφής (destructor)

Χρησιμοποιούνται για την “καταστροφή” ενός στιγμιότυπου μιας κλάσης (αντικειμένου) και την αποδέσμευση της μνήμης που δεσμεύτηκε κατά την δημιουργία του.

- Η συνάρτηση καταστροφής έχει το ίδιο όνομα με τη κλάση αλλά πριν από το όνομα υπάρχει ο χαρακτήρας ~
- Κάθε κλάση μπορεί να περιέχει μία συνάρτηση καταστροφής
- Δεν έχει παραμέτρους (ορίσματα)
- Οπως και η συνάρτηση κατασκευής δεν έχει τύπο επιστρεφόμενης τιμής.

Constructors-Destructors 1

```
class MyString
{
private:
  char *m_string_p;
  int str_length;

public:
  MyString(const char *string_p="")    //consructor
  {
    str_length = strlen(string_p) + 1; // Length string + 1 for terminator
    m_string_p = new char[str_length];   // Allocate a buffer
    strncpy(m_string_p, string_p, str_length);
    m_string_p[str_length-1] = '\0';    // Make sure the string is terminated
  }

  ~MyString() // destructor
  {
      delete[] m_string_p;       // We need to deallocate our buffer
      m_string_p = 0;           // Set m_pchString to null just in case
  }
  char* GetString() { return m_string_p; }
  int GetLength() { return str_length; }
};
```

Constructors-Destructors 1

```
int main()
{
    MyString cMyName("Alex");
    std::cout << "My name is: " <<
cMyName.GetString() << std::endl;
    return 0;
} // cMyName destructor called here!
```

Constructors-Destructors 2

```
#include <iostream>
using namespace std;
const unsigned MIN_SIZE=2;
class DArray
{
  private:
    double *ArrayPtr;
    unsigned ArraySize;
  public:
    DArray(unsigned size=MIN_SIZE)
    {
      ArraySize = (size<MIN_SIZE)? MIN_SIZE:size;
      ArrayPtr = new double[ArraySize];
    }
    ~DArray() {delete [] ArrayPtr};

    unsigned getArraySize() const { return ArraySize; }

    void store(double x, unsigned index)
    { ArrayPtr[index] = x;}

    double recall(unsigned index)
    { return ArrayPtr[index]; }
};
```

Constructors-Destructors 2

```
int main()
{
   DArray Arr(10);
   double x;

   for (unsigned i=0;i<Arr.getArraySize();i++)
    {
      x=double(i);
      x = x*x -5*x +10;
      Arr.store(x,i);
    }

   for (i=0; i<getArraySize();i++)
      cout << "Arr["<<i<<"] = " << Arr.recall(i)<<endl;

   return 0;
}
```

Στατικά μέλη κλάσης (Static members)

Μια κλάση μπορεί να έχει στατικά μέλη, δεδομένα ή συναρτήσεις

Οι τιμές των στατικών μεταβλητων είναι ίδιες για όλα τα αντικείμενα (στιγμιότυπα) αυτής της κλάσης.

Αν η στατική μεταβλητή είναι private δεν μπορεί να χρησιμοποιηθεί μια public συνάρτηση για να προσπελασθεί η μεταβλητή.

Μια στατική συνάρτηση δεν συνδέεται με τα αντικείμενα μιας κλάσης και μπορεί να κληθεί πρίν δηλωθεί ένα αντικείμενο.

Η στατική συνάρτηση έχει πρόσβαση μόνο στις στατικές μεταβλητές

```
class Box
{
 private:
  double m_length,m_width,m_height;
  static int BoxNumber;
  int BoxID;
 public:

  Box(double length=1.0,double width=1.0,double height=1.0)
  {
    m_length = length;
    m_width  = width;
    m_height = height;
    BoxID = BoxNumber++;
  }
  void BoxDimentions(double length,double width,double height)
  {
     m_length = length;
     m_width = width;
     m_height = height;
  }
  double BoxVolume()
  { return m_length * m_width * m_height ;  }
  int GetBoxID()
  { return BoxID; }

};
```

```
int Box::BoxNumber = 1;

int main()
{
    Box Box1;
    Box Box2;
    Box Box3;

    cout << Box1.GetBoxID() << "   "<<Box2.GetBoxID()<<"   "
         <<Box3.GetBoxID() << endl;
    return 0;
}
```

Δείκτες σε κλάσεις - Pointers to classes

```
int main()
{
  Box Box1,*Box2,*Box3;
  Box *Box_array = new Box[2];
  Box2 = new Box;
  Box3 = &Box1;

  Box1.BoxDimentions(1,2,3);
  Box2->BoxDimentions(10,20,30); //
  Box_array[1].BoxDimentions(4,5,6)

  cout << "*Box2 Volume = " <<Box2->BoxVolume() << endl;
  cout <<"Box_array[0] Volume = "<<Box_array[0].BoxVolume() << endl;
  cout <<"Box_array[1] Volume = "<<Box_array[1].BoxVolume() << endl;

  delete Box2;
  delete[] Box_array;
  return 0;
}
```

-> Τελεστής μέλους για δείκτες

`this` pointer

- Είναι ένας δείκτης που δημιουργεί αυτόματα ο compiler της C++.
- Δείχνει στη διεύθυνση ενός αντικειμένου, στιγιοτύπου μιας κλάσης.
- Ο δείκτης `this` προστίθεται στις παραμέτρους μίας συνάρτησης μέλους μιας κλάσης και δείχνει στο αντικείμενο για το οποίο έγινε η κλήση της συνάρτησης.
  - Έστω μία συνάρτηση μέλος `fun(int a) {}`
  - Η κλήση της θα είναι `fun(&object, int a)`
- Ο δείκτης `this` δεν είναι μέλος του αντικειμένου και δεν συμπεριλαμβάνεται στη μνήμη που δεσμεύει το αντικείμενο και δεν φαίνεται στο μέγεθος που επιστρέφει η `sizeof`
- Δεν ισχύει για στατικές συναρτήσεις

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

class Line
{
  private:
    double start_x,start_y,end_x,end_y;
  public:
    Line (double,double,double,double);
    Line& SetLineStart(double,double );
    Line& SetLineEnd(double,double );
    double LineLength();
    void print();
    int compare(Line);
};

Line::Line (double start_x=0.0,double start_y=0.0,
            double end_x=1.0, double end_y=1.0)
{
  this -> start_x = start_x;
  this -> start_y = start_y;
  this -> end_x = end_x;
  this -> end_y = end_y;
}
double Line::LineLength(double start_x,double start_y,
                        double end_x, double end_y)
{
```

```
Line& Line::SetLineStart(double x,double y)
{
  start_x = x;
  start_y = y;
  return *this;
}

Line& Line::SetLineEnd(double x,double y)
{
  end_x = x;
  end_y = y;
  return *this;
}
void Line::print()
{
    cout << "Line Start Point : ( " << this -> start_x <<","<<
this -> start_y << ")" << endl;
    cout << "Line End Point   : ( " << end_x <<","<< end_y <<
")" << endl;
}

int Line::compare(Line line)
{
    if (this -> LineLength() > line.LineLength())
        return 1;
    else if (this -> LineLength() == line.LineLength())
        return 0;
    else
        return -1;
```

```
int main()
{
  Line line1,line2;

  //line1.SetLineStart(a,b)
  //line1.SetLineStart(&line1,a,b)

  line1.SetLineStart(1.5,1.5).SetLineEnd(2.5,2.5);
  // διαδοχική κλήση συναρτήσεων
  // Πρώτα εκτελείται η line1.SetLineStart(1.5,1.5) και επιστρέφει μια
  // αναφορά στο αντικείμενο line1. Μετά εκτελείται η line1.SetLineEnd(2.5,2.5)

  line2.SetLineStart(3.0,3.0).SetLineEnd(5.0,5.0);

  line1.print();
  cout << "Line 1 length :" << line1.LineLength() << endl;
  line2.print();
  cout << "Line 2 length :" << line2.LineLength() << endl;

  if (line1.compare(line2) == 0)
    cout << " Line 1 = Line 2" << endl;
  if (line1.compare(line2) == 1)
    cout << " Line 1 > Line 2" << endl;
  if (line1.compare(line2) == -1)
    cout << " Line 1 < Line 2" << endl;

  return 0;
}
```

κλήση της συνάρτησης

τροποποίηση της κλήσης από τον compiler

Φιλικές Συναρτήσεις - Φιλικές Κλάσεις

- Μια συνάρτηση που δεν είναι μέλος μιας κλάσης μπορεί να δηλωθεί μέσα στη κλάση ως φιλική με τη λέξη `friend`
- Η φιλική συνάρτηση αν και δεν βρίσκεται στην εμβέλεια της κλάσης έχει πρόσβαση στις ιδιωτικές μεταβλητές της
- Μπορούν να δηλωθούν και φιλικές κλάσεις

```
class Class_A
{
 ......................
 friend class Class_B
 ......................
}
```

```
class Box
{
 private:
  double m_length,m_width,m_height;
 public:

  Box(double length=1.0,double width=1.0,double height=1.0)
  {
    m_length = length;
    m_width  = width;
    m_height = height;
  }
  void BoxDimentions(double length,double width,double height)
  {
     m_length = length;
     m_width = width;
     m_height = height;
  }
  double BoxVolume()
  { return m_length * m_width * m_height ;  }

  friend bool EqualBoxes(const Box &Box_A,const Box &Box_B);

};
```

```
bool EqualBoxes(const Box &Box_A,const Box &Box_B)
{
    //const : η συνάρτηση δεν μπορεί να αλλάξει τις τιμές των
    //        μεταβλητών  των αντικειμένων
    return (Box_A.m_length == Box_B.m_length)&&
           (Box_A.m_width == Box_B.m_width)&&
           (Box_A.m_length == Box_B.m_length);
}


int main()
{
    Box Box1(1.0,2.0,3.0);
    Box Box2(1.0,2.0,3.0);
    Box Box3(5.0,2.0,3.0);

    cout << "Box1 equal Box2 ? " << EqualBoxes(Box1,Box2) <<endl;
    cout << "Box1 equal Box3 ? " << EqualBoxes(Box1,Box3) <<endl;

    return 0;
}
```

Υπερφόρτωση τελεστών
Overloading operators

```
main
{
  int a,b,c;
  double x,y,z;
  int array1[10],array2[10],array3[10];
  Box Box1,Box2,Box3;

  a = b+c;
  x = y+z;


  array1 = array2+array3; // ????
  Box1 = Box2+Box3;       // ????
}
```

Έχει οριστεί ο κανόνας της πρόσθεσης

Δεν έχει οριστεί ο κανόνας της πρόσθεσης

a+b ⟶ operator+(a , b)

Υπερφόρτωση τελεστών για διανύσματα

```
#include <iostream>
using namespace std;
class IVector
{
  public:
  int x,y;
  IVector () {};
  IVector (int,int);
  IVector operator + (IVector);
};

IVector::IVector (int a, int b)
{
  x = a;
  y = b;
}
IVector IVector::operator+ (IVector param)
{
  IVector temp;
  temp.x = x + param.x;
  temp.y = y + param.y;
  return (temp);
}
```

```
int main ()
{
  IVector a (3,1);
  IVector b (1,2);
  IVector c;
  c = a + b;
  cout << c.x << ","
       << c.y;
  return 0;
}
```

Πρόσθεση διανυσμάτων

Υπερφόρτωση τελεστών – χρήση του δείκτη `this`

```
#include <iostream>
using namespace std;
class IVector
{
  public:
  int x,y;
  IVector () {};
  IVector (int,int);
  IVector operator + (IVector&);
};

IVector::IVector (int a, int b)
{
  x = a;
  y = b;
}
IVector IVector::operator+ (IVector &param)
{
  x = x + param.x;
  y = y + param.y;
  return *this;
}
```

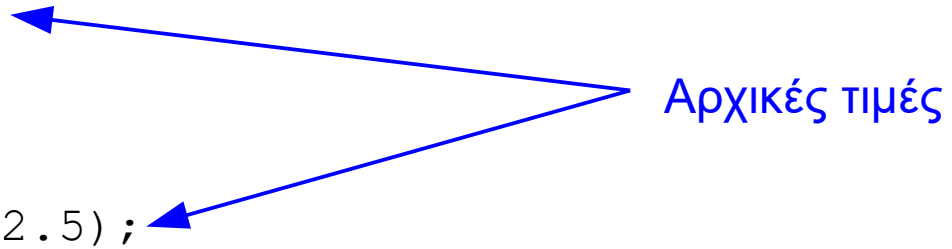
Πρόσθεση διανυσμάτων

Υπερφόρτωση πρόσθεσης διανυσμάτων για ακέραιους και πραγματικούς

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

class IVector
{
  public:
    int x,y;
    IVector () {};
    IVector (int,int);
    IVector operator + (IVector);
};

IVector::IVector (int a, int b)
{   x = a;  y = b; }

IVector Ivector::operator+
        (IVector param)
{
    IVector temp;
    temp.x = x + param.x;
    temp.y = y + param.y;
    return (temp);
}
```

```
class RVector
{
  public:
    double x,y;
    RVector () {};
    RVector (double,double);
    RVector operator + (RVector);
};

RVector::RVector (double a, double b)
{   x = a;  y = b; }

RVector Rvector::operator+
        (RVector param)
{
    RVector temp;
    temp.x = x + param.x;
    temp.y = y + param.y;
    return (temp);
}
```

```
int main ()
{
    IVector a (3,1);
    IVector b (1,2);
    IVector c;

    RVector rx (1.3,2.5);
    RVector ry (5.5,4.8);
    RVector rz;

    c = a + b;
    cout << c.x << "," << c.y << endl;

    rz = rx + ry;
    cout << rz.x << "," << rz.y << endl;

    return 0;
}
```



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

class C_Complex
{
  public:
   double real,imag;
   C_Complex(double , double );
   C_Complex operator+ (C_Complex);
   C_Complex operator- (C_Complex);
   C_Complex operator* (C_Complex);

   friend ostream& operator<<(ostream & os,C_Complex &c);
   friend istream& operator>>(istream & is,C_Complex &c);
};
C_Complex::C_Complex(double real_part=0.0, double imag_part=0.0)
{
   real=real_part;
   imag=imag_part;
}
C_Complex C_Complex::operator+ (C_Complex param)
{
   C_Complex tmp;
   tmp.real = real + param.real;
   tmp.imag = imag + param.imag;
   return tmp;
```

Αριθμητικοί τελεστές

Τελεστές εισόδου εξόδου

```
C_Complex C_Complex::operator- (C_Complex param)
{
    C_Complex tmp;
    tmp.real = real - param.real;
    tmp.imag = imag - param.imag;
    return tmp;
}

C_Complex C_Complex::operator* (C_Complex param)
{
    C_Complex tmp;
    tmp.real = real * param.real - imag * param.imag;
    tmp.imag = real * param.imag - imag * param.real;
    return tmp;
}

ostream& operator<<(ostream & os,C_Complex &c)
{
    os << "(" << c.real << " + i" << c.imag << ")";
    return os;
}

istream& operator>>(istream & is,C_Complex &c)
{
    cout << "Real part= ";is >> c.real;
    cout << "Imag part= ";is >> c.imag;
    return is;
}
```

```
int main()
{
    C_Complex x(1.0,2.0);
    C_Complex y(3.0,4.0);
    C_Complex z,w;

    z = x + y;

    cin >> w;
    cout << x << "+" << y << "=" << z << endl;

    cout << " w= "<<w<<endl;
/*
    cout << "(" << x.real << " + i" << x.imag << ")";
    cout << " + ";
    cout << "(" << y.real << " + i" << y.imag << ")";
    cout << " = ";
    cout << "(" << z.real << " + i" << z.imag << ")";
    cout << endl;
*/
    z = x - y;
    return 0;
}
```

Εκφραση	Τελεστής	Συνάρτηση μέλος	Γενική συνάρτηση
@a	+ - * & ! ~ ++ --	A::operator@()	operator@(A)
a@	++ --	A::operator@(int)	operator@(A,int)
a@b	+ - * / % ^ & | < > == != >= <= >> << && || ,	A::operator@(B)	operator@(A,B)
a@b	= += -= *= /= %= ^= &= |= <<= >>= []	A::operator@(B)	-
a(b,c,...)	()	A::operator@(B...)	-
a->x	->	A::operator@->()	-

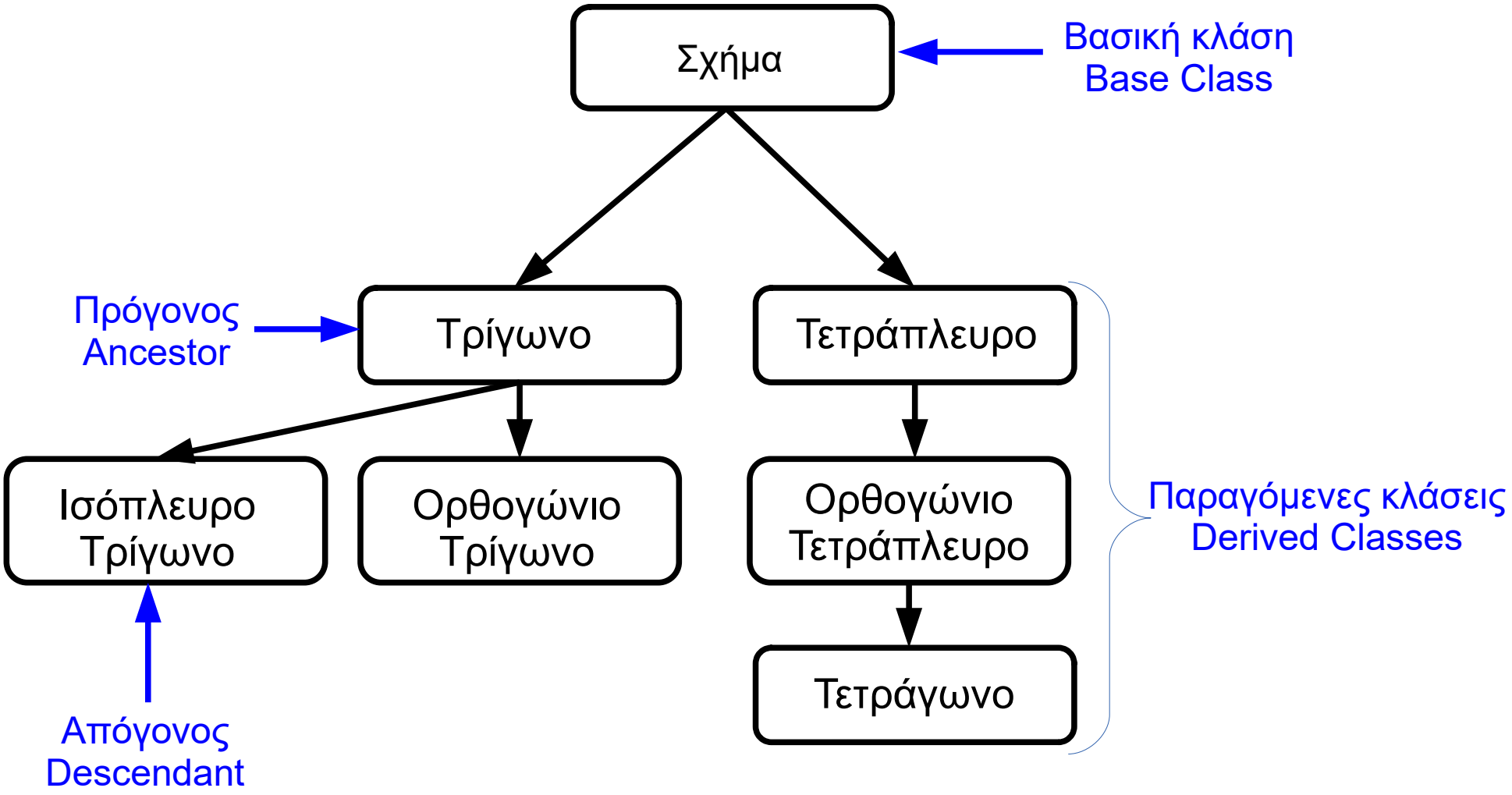
a,b,c αντικείμενα κλάσης A,B,C αντίστοιχα

```
class IntArray
{
private:
    int ArrayLength, *ArrayData;
public:
    IntArray()
    {
        ArrayLength = 0;
        ArrayData = 0;
    }
    IntArray(int Length)
    {
        ArrayData = new int[Length];
        ArrayLength = Length;
    }
    ~IntArray()
    { delete[] ArrayData; }
    void Erase()
    {
        delete[] ArrayData;
        ArrayData = 0;
        ArrayLength = 0;
    }
     int& operator[](int Index)
    {
        assert(Index >= 0 && Index < ArrayLength);
        return ArrayData[Index];
    }
     int GetLength() { return ArrayLength; }
```

Κληρονομικότητα - Inheritance
Παραγόμενες κλάσεις - Derived classes

- Μια κλάση μπορεί να παράγεται (να είναι απόγονος-descendant) από μια άλλη κλάση (γονέα-parent) και να *κληρονομεί* μερικά (ή όλα) από τα μέλη της κλάσης-γονέα (δεδομένα ή συναρτήσεις) μαζί με τα δικά της.
- Η αρχική κλάση στην ιεραρχία ονομάζεται **βασική κλάση** (**base class)**.
- Η παραγόμενη κλάση μπορεί να επαναορίσει συναρτήσεις των κλάσεων προγόνων.
- Οι συναρτήσεις κατασκευής και καταστροφής δεν κληρονομούνται. Κατά τη κατασκευή αντικειμένου καλείται πρώτα η συνάρτηση της βασικής κλάσης.
- Οι φιλικές συναρτήσεις δεν κληρονομούνται γιατί δεν είναι μέλη της κλάσης.
- Προσδιοριστικό protected. Η παραγόμενη κλάση έχει πρόσβαση στα προστατευμένα μέλη της κλάσης γονέα.

Class Hierarchy



```
class ονομα κλάσης : [public] κλάση γονέας
{                  // ή private ή protected σχέση κληρονομικότητας

  <φιλικές κλάσεις>
  private: //πρόσβαση μόνο από τις συναρτήσεις μέλη
           //της παρούσας κλάσης
    <δεδομένα>
    <συναρτήσεις κατασκευής>
    <συναρτήσεις καταστροφής>
  protected:// πρόσβαση και απο τις συναρτήσεις μέλη
            // των απογόνων (παραγόμενων κλάσεων)
    <δεδομένα>
    <συναρτήσεις κατασκευής>
    <συναρτήσεις καταστροφής>
  public:
    <δεδομένα>
    <συναρτήσεις κατασκευής>
    <συναρτήσεις καταστροφής>
    <συναρτήσεις μέλη>

    <φιλικές συναρτήσεις>
};
```

Σχέση κληρονομικότητας

Καθορίζει τα δικαιώματα πρόσβασης των μελών που κληρονομεί μια κλάση

- **Public:** Διατηρούνται τα ίδια δικαιώματα
- **Protected:** Όλα τα δημόσια μέλη κληρονομούνται ως προστατευμένα.
- **Private:** Όλα τα μέλη κληρονομούνται ως ιδιωτικά

```
class A_class
{private  :int a1;
 protected:int a2;
 public   :int a3;
};
```

```
class B_class: public A_class
            // ή private ή protected
{private  :int b1;
 protected:int b2;
 public   :int b3;
};
```

```
int main()
{
 A_class A1;
 B_class B1;
```

Ο,τι δικαιώματα ορίζει η A_class

```
A1.a1 = ;
A1.a2 = ;
A1.a3 = ;
```

Ο,τι δικαιώματα ορίζει η B_class

```
B1.b1 = ;
B1.b2 = ;
B1.b3 = ;
```

```
B1.a1 = ;
B1.a2 = ;
B1.a3 = ;
}
```

	Προσδιοριστικό πρόσβασης		
	public	protected	private
B1.a1	private	private	private
B1.a2	protected	protected	private
B1.a3	public	protected	private

Και για τις κλάσεις Παράγωγα τη B_class

Παραδειγμα κληρονομικοτητας

```
include <iostream>
include <math.h>

cosnt double pi = 4 * atan(1);
inline double sqr(double x) { return x*x;}

using namespace std;

class cCircle //Βασική κλάση
{
   protected:
     double radius;

   public:
     cCircle(double radius_value=0): radius(radius_value) {}
     void set_radius(radius_value) {radius=radius_value;}
     double get_radius() const {return radius;}
     double area() const { return pi*radius*radius;}
     void show_data();
};
```

```
class cCylinder : public cCircle //παραγόμενη κλάση
{
  protected:
    double height;
  public:
    cCylinder(double radius_value=0,double height_value=0)
      :height cCircle(radius_value),(height_value) {}
    void set_height(double height_value)
      {height = height_value;}
    double get_height() const {return height;}
    double area() const
      { return 2*cCircle::area()+2*pi*radius*height;}
    void show_data();
};

void cCircle::show_data()
{
  cout << "Circle radius    = "<< get_radius() <<endl
       << "Circle area      = "<< area() << endl << endl;
}

void cCylinder::show_data()
{
  cout << "Cylinder radius  = "<< get_radius() << endl
       << "Cylinder height  = "<< get_height() << endl
       << "Cylinder area    = "<< area() << endl << endl;
}
```

```
int main()
{
  cCircle Circle(1);
  cCylinder(10 ,1);

  Circle.show_data();
  Cylinder.show_data();
  return 0;
}
```

class cCircle //Βασική κλάση
class cCylinder : public cCircle //παραγόμενη κλάση
dynamic_cast Χρησιμοποιείται σε δείκτες και αναφορές σε κλάσεις

```
cCirle * pba = new cCylinder ;
cCircle * pbb = new  cCircle;
cCylinder *pd;
pd = dynamic_cast< cCylinder *> pba;
if (pd==0) cout << "Null pointer on first type-cast.\n";
```

pd = 0 : pba δείχνει σε ένα αντικείμενο παραγόμενης (πλήρους) κλάσης

```
pd = dynamic_cast<cCylinder *> pbb;
if (pd==0) cout << "Null pointer on second type-cast.\n";
```

pd ≠ 0 : pba δείχνει σε ένα αντικείμενο βασικής (μη πλήρους) κλάσης

Πολλαπλή κληρονομικότητα

```
#include <iostream>
using namespace std;
class CPolygon
{
  protected:
    int width, height;
  public:
  void set_values (int a, int b)
   { width=a; height=b;}
};
class Coutput
{
  public:
    void output (int i) {cout << i << endl;}
};

class CRectangle: public CPolygon, public Coutput
{
  public:
    int area (){ return (width * height); }
};
class CTriangle: public CPolygon, public Coutput
{
  public:
   int area (){ return (width * height / 2); }
};
```

```
int main ()
{
  CRectangle rect;
  CTriangle trgl;
  rect.set_values (4,5);
  trgl.set_values (4,5);
  rect.output (rect.area());
  trgl.output (trgl.area());
  return 0;
}
```

Πολυμορφισμός

Αντικείμενα που ανήκουν σε παρόμοιες κλάσεις μπορούν να έχουν κοινό τρόπο προσπέλασης, με αποτέλεσμα ο χρήστης να μπορεί να τα χειριστεί με τον ίδιο τρόπο χωρίς να χρειάζεται να μάθει νέες διαδικασίες

Δείκτες (pointers) και αναφορές (references) στη βασική κλάση

Ενας δείκτης σε μια παραγόμενη κλάση και ένας δείκτης στη βασική κλάση έχουν συμβατούς τύπους

Δείκτες στη βασική κλάση
pointers to base class

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

class Cpolygon
{
  protected:
    int width, height;
  public:
    void set_values (int a, int b)
    { width=a; height=b; }
};
class CRectangle: public Cpolygon
{
  public:
    int area ()
   { return (width * height); }
};
class CTriangle: public Cpolygon
{
  public:
    int area ()
      { return (width * height / 2); }
};
```

Οι κλάσεις `Crectangle` και `CTriangle` ειναι παράγωγα της `Cpolygon` και κληρονομούν τα μέλη `width, height` και `set_values`

Δύο διαφορετικές υλοποιήσεις της συνάρτησης `arrea`

```
int main ()
{
  CRectangle rect;
  CTriangle trgl;

  CPolygon *ppoly1 = &rect;
  CPolygon *ppoly2 = &trgl;

  CPolygon &ppoly3 = rect;
  CPolygon &ppoly4 = trgl;

  ppoly1->set_values (4,5);
  ppoly2->set_values (4,5);

  cout << rect.area() << endl;
  cout << trgl.area() << endl;

  ppoly1.set_values (10,20);
  ppoly2.set_values (10,20);
  cout << rect.area() << endl;
  cout << trgl.area() << endl;

  return 0;
}
```

ppoly1,ppoly2:
Δείκτες σε αντικείμανα κλάσης Cpolygon που "*δείχνουν*" στα στιγμιότυπα rect και trgl αντίστοιχα.
ppoly3,ppoly4:

Οι δείκτες ppoly1 και ppoly2 αναφέρονται σε μέλος της Cpolygon που κληρονομούν οι Crectangle και Ctriangle

Η area δεν ορίζεται στην Cpolygon και δεν μπορεί να χρησιμοποιηθεί δείκτης

Virtual functions (members)

```
class CPolygon
{
  protected:
    int width, height;
  public:
    void set_values (int a, int b)
      {width=a; height=b;}

    virtual int area ()
      {return (0);}
};
class CRectangle: public Cpolygon
{
  public:
    virtual int area ()
      {return (width * height);}
};
class CTriangle: public CPolygon
{
  public:
    virtual int area ()
      {return (width * height / 2);} }
};
```

Virtual function
Η `area` είναι μέλος της `Cpolygon`.
Είναι δυνατή η προσπέλαση με δείκτη στη βασική κλάση

Οι παραγόμενες κλάσεις επαναορίζουν την `area`
Η λέξη `virtual` είναι προαιρετική.

Μια κλάση που περιέχει τη δήλωση μιας virtual function ή κληρονομεί μια virtual function ονομάζεται πολυμορφική

Virtual functions (members)

```
int main ()
{
  CRectangle rect;
  CTriangle trgl;
  CPolygon poly;

  CPolygon * ppoly1 = &rect;
  CPolygon * ppoly2 = &trgl;
  CPolygon * ppoly3 = &poly;

  ppoly1->set_values (4,5);
  ppoly2->set_values (4,5);
  ppoly3->set_values (4,5);

  cout << ppoly1->area() << endl;
  cout << ppoly2->area() << endl;
  cout << ppoly3->area() << endl;

  return 0;
}
```

Κλήση της `area` με χρήση δείκτη στη βασική κλάση

- Οι virtual συναρτήσεις προσφέρουν ευκολίες στη σχεδίαση επεκτάσιμων συστημάτων
- Μια συνάρτηση που δηλώνεται virtual διατηρεί αυτή την ιδιότητα σε όλες τις κλάσεις της ιεραρχίας
- Οι παραγόμενες κλάσεις μπορούν να επαναορίσουν τη virtual συνάρτηση ή να την διατηρήσουν
- Οι συναρτήσεις κατασκευής δεν μπορεί να είναι virtual.

Αφηρημένες κλάσεις
Abstract base classes - Pure Virtual Classes

- Η C++ μας δίνει τη δυνατότητα να ορίσουμε μια γενική δομή κάποιων κλάσεων, χωρίς όμως να δώσουμε μια ακριβή υλοποίηση.

- Αφηρημένη είναι μια κλάση που έχει ως μέλος μια τουλάχιστον “pure virtual” συνάρτηση.
  - Pure virtual είναι η συνάρτηση που έχει όνομα και τύπο αλλά δεν δίνεται ο ορισμός της (δεν υπάρχουν οι εντολές της συνάρτησης - { εντολές }).

```
virtual int p_v_function() = 0;
```

- Δεν μπορούμε να δηλώσουμε αντικείμενα μιας αφηρημένης κλάσης αλλά μπορούμε να δηλώσουμε δείκτες σε αφηρημένες κλάσεις.

- Χρησιμοποιούνται ως βασικές κλάσεις (base classes) σε ιεραρχίες κληρονομικότητας.

Παράδειγμα αφηρημένης κλάσης - 1

```
#include <iostream>
using namespace std;
class CPolygon
{
  protected:
    int width, height;
  public:
    void set_values (int a, int b)
    { width=a; height=b; }
    virtual int area (void) =0;
};
class CRectangle: public CPolygon
{
  public:
    int area (void)
    { return (width * height); }
};
class CTriangle: public CPolygon
{
  public:
    int area (void)
    { return (width * height / 2); }
};
```

Abstract base class

Pure virtual function `area`

- Δεν υπάρχει ορισμός
- Έχει πρόσβαση στις μεταβλητές της κλάσης

Κάθε κλάση ορίζει τη δική της συνάρτηση `area`

Παράδειγμα αφηρημένης κλάσης - 2

```
int main ()
{
   CRectangle rect;
   CTriangle trgl;

   CPolygon * ppoly1 = &rect;
   CPolygon * ppoly2 = &trgl;


   ppoly1->set_values (4,5);
   ppoly2->set_values (4,5);


   cout << ppoly1->area() << endl;
   cout << ppoly2->area() << endl;
   return 0;
}
```

Δείκτες αφηρημένων αντικειμένων τύπου `Cpolygon` που δείχνουν σε συγκεκριμένα αντικείμενα `Crectangle` και `CTriangle`

Τιμές για ορθογώνιο

Τιμές για τρίγωνο

Κλήση της συνάρτησης `area` για ορθογώνιο

Κλήση της συνάρτησης `area` για τρίγωνο

Interface classes

- Δεν έχει ως μέλη μεταβλητές
- Εχει μέλη μόνο pure virtual functions

Πρότυπα Templates

Η υπερφόρτωση μας επιτρέπει να έχουμε συναρτήσεις για πολλαπλούς τύπους παραμέτρων αλλά πρέπει να γράψουμε μία συνάρτηση για κάθε τύπο.

Πρότυπα συναρτήσεων Function Templates

- Ειδικές συναρτήσεις με παραμέτρους και επιστρεφόμενη τιμή γενικούς τύπους στοιχείων ή κλάσεων.
- Ένας ορισμός της συνάρτησης αρκεί για όλους τους τύπους.
- Τα ορίσματα και η επιστρεφόμενη τιμή δεν είναι τιμές ενός τύπου στοιχείων αλλά παράμετροι τύπων (Template Parameters)

Μορφή δήλωσης:

```
template <class όνομα_τύπου>
όνομα_τύπου όνομα_συνάρτησης(όνομα_τύπου παράμετρος,...)
{    }
template <typename όνομα_τύπου>
όνομα_τύπου όνομα_συνάρτησης (όνομα_τύπου παράμετρος,...)
{    }
```

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

template <class T1>

void swap_t( T1 &a , T1 &b)
{
    T1 temp;
    temp = a;
    a = b;
    b = temp;
}
int main()
{
    int a=10,b=20;
    float x=1.1,y=2.2;
    char c1='A',c2='Z';

    cout << a << "   " << b << endl;
    swap_t(a,b);
    cout << a << "   " << b << endl << endl;
    cout << x << "   " << y << endl;
    swap_t(x,y);
    cout << x << "   " << y << endl << endl;
    cout << c1 << "   " << c2 << endl;
    swap_t(c1,c2);
    cout << c1 << "   " << c2 << endl << endl;
    Return 0;
}
```

Ένας ορισμός της `swap_t`

Κλήσεις της `swap_t` με διαφορετικούς τύπους στοιχείων

```
template <typename T2>
T2 GetMax (T2 a, T2 b)
{
    T2 result;
    result = (a>b)? a : b;
    return result;
}

int main()
{
   int a=10,b=20,m;
   float x=1.5,y=3.5,z;
   char c1='A',c2='Z',c3;

   c = GetMax(a,b);
   z = GetMax(x,y);
   c3 = GetMax(c1,c2);

   //z = GetMax<float>(x,y);
   //c = GetMax<int>(a,b);
   //c3 = GetMax<char>(c1,c2);

   cout .....;
   return 0;
}
```

Δεν είναι απαραίτητο

Πρότυπα συναρτήσεων

- Οι συναρτήσεις swap_t και GetMax έχουν παραμέτρους του ίδιου τύπου και η GetMax επιστρέφει τιμή του ίδιου τύπου με τις παραμέτρους της

```
int a=10,b=20,c;
float x=1.5,y=3.5,z;
z = GetMax(a,y);
```

Λάθος: `a` και `y` και `z` δεν είναι του ίδιου τύπου

- Μπορούμε να ορίσουμε πρότυπα συναρτήσεων με παραμέτρους διαφορετικού τύπου.
  - Πρέπει να ορίσουμε περισσότερες παραμέτρους τύπων (template parameters)
  - `template <typename T1, typename T2, ... >`
  - `template <class T1, class T2, ... >`

Πρότυπα συναρτήσεων

```
template <typename T3, typename T4>
T4 GetMax (T3 a, T4 b)
{
    T4 result;
    result = (a>b)? a : b;
    return result;
}
int main()
{
   int a=10,b=20,m;
   float x=1.5,y=3.5,z;
   char c1='A',c2='Z',c3;

   z = GetMax(a,x);
   c3 = GetMax(c1,a);
   cout .....;
   return 0;
}
```

Υπερφόρτωση προτύπων συναρτήσεων

```
template <typename T2>
T2 GetMax (T2 a, T2 b)
{
    T2 result;
    result = (a>b)? a : b;
    return result;
}
template <typename T3, typename T4>
T4 GetMax (T3 a, T4 b)
{
    T4 result;
    result = (a>b)? a : b;
    return result;
}
int main()
{
   int a=10,b=20,m;
   float x=1.5,y=3.5,z;
   char c1='A',c2='Z',c3;
   c = GetMax(a,b);
   z = GetMax(a,x);
   c = GetMax(a,c1);

   return 0;
}
```

Δύο παράμετροι του ίδιου τύπου

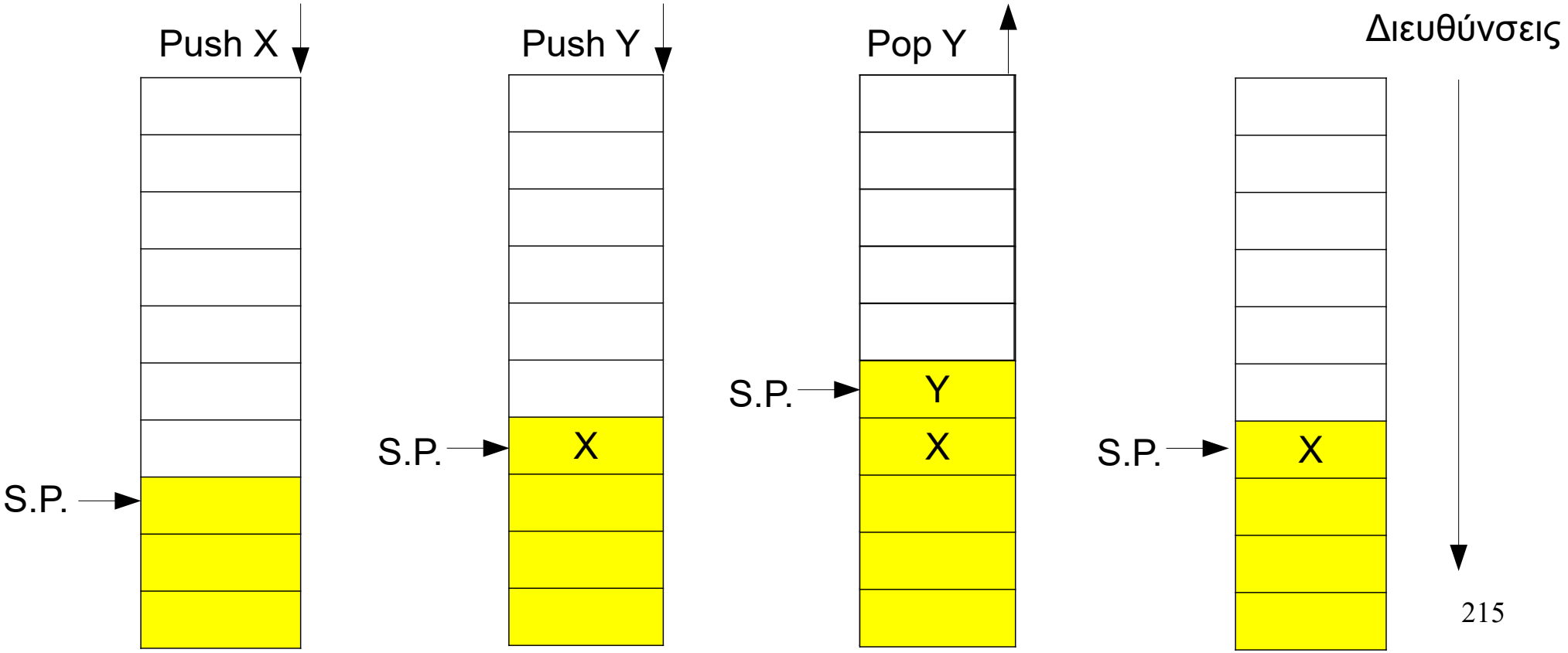
Υπερφόρτωση της `GetMax`

Δύο παράμετροι διαφορετικών τύπων

Στοίβα - Δείκτης στοίβας
Stack – Stack Pointer (SP)

PUSH : Μείωση της τιμής του SP ώστε να δείχνει στη προηγούμενη (κενή) θέση
Αποθήκευση της τιμής στη νέα κορυφή της Stack.

POP : Διάβασμα της τιμής από τη κορυφή της Stack
Αύξηση της τιμής του SP ώστε να δείχνει στη νέα κορυφή της Stack



Πρότυπα κλάσεων Class Templates

Stack για διάφορους τύπους στοιχείων

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

template< typename T >
class Stack
{
  private:
    int stack_size;
    int StackPointer;
    T *StackArray;
  public:
    Stack( int = 10 );
    ~Stack()
    {
       delete [] StackArray;
       StackArray = 0;
    }
    bool push( const T& );
    bool pop( T& );
    bool isEmpty() const
    { return StackPointer == stack_size; }
    bool isFull() const
    { return StackPointer == 0; }

    int GetStackPointer();
```

Συναρτήσεις μέλη της κλάσης stack

```
template< typename T >
Stack< T >::Stack( int s )//πρότυπο συνάρτηση κατασκευής της κλάσης
  :stack_size(s > 0 ? s : 20),          //Έλεγχος μεγέθους, μέγιστο 20
   StackPointer(stack_size),                        //stack κενή
   StackArray(new T[stack_size] )   //δέσμευση μνήμης
{  }
template< typename T >
bool Stack< T >::push(const T &pushValue)//εισαγωγή στοιχείου στη stack
{
  if ( !isFull() )
  {
    StackArray[ --StackPointer ] = pushValue;// ενημέρωση του δείκτη
    return true;// εισαγωγή επιτυχής
  }
  return false;//  εισαγωγή ανεπιτυχής
}
template< typename T >
bool Stack< T >::pop( T &popValue )//αφαίρεση  στοιχείου από τη stack
{
  if ( !isEmpty() )
  {
     popValue = StackArray[ StackPointer++ ];// ενημέρωση του δείκτη
     return true;// αφαίρεση επιτυχής
  }
  return false;  // αφαίρεση ανεπιτυχής
}
```

```
template< typename T >
int Stack<T> :: GetStackPointer()
{
  return StackPointer;
}


template< typename T >
int StackFunction( Stack< T > StackType) //Λειτουργίες της stack
{
   int PushPop;
   T StackValue;
   do
   {
       cout << " Enter 1(PUSH), 2(POP), 3(Exit) ";
       if (!(cin >> PushPop))
       {
          cout << " ERROR ! Enter Integer "<< endl;
          cin.clear();
          cin.ignore();
       }
       cout << endl;
```

```
  switch (PushPop)   //Επιλογή λειτουργίας
  {
    Case 1:           //PUSH
    {
      cout << "Enter an element to PUSH into Stack : ";
      cin >>StackValue;
      cout << endl;
      if(!StackType.push(StackValue ))
         cout <<"Stack is full. Cannot push "<<StackValue<<endl;
      else
         cout <<"Element "<<StackValue <<"  Into Stack position "
              << StackType.GetStackPointer() << endl;
      break;
    }
    Case 2:         //POP
    {
      if(!StackType.pop( StackValue ))
        cout << endl << "Stack is empty. Cannot pop"<< endl;
      else
        cout << "Element "<<StackValue << "  From Stack position "
             << StackType.GetStackPointer() << endl;
      break;
    }
    case 3: return 0;  //Έξοδος
  }
}while(true );
```

```
int main()
{
  unsigned ElementType;
  unsigned Number_Of_Elements;

  do
  {
    cout << " Stack Type 1(Int) 2(Double) 3(Char) ";
    if (!(cin >> ElementType))
       {
          cout << endl << endl << " ERROR ! Enter Integer "<< endl;
          cin.clear();
          cin.ignore();
       }
    cout <<endl;
  }while((ElementType!= 1)&&(ElementType!= 2)&&(ElementType!= 3));

  cout << "Number of Stack Elements : ";
  cin >> Number_Of_Elements;
```

```
switch (ElementType)
  {
     case 1:
         { Stack< int > intStack(Number_Of_Elements);
           StackFunction(intStack);
           intStack.~Stack();
           return 0;
           break;
         }
     case 2:
         {
           Stack< double > doubleStack(Number_Of_Elements);
           StackFunction(doubleStack)
           doubleStack.~Stack();
           return 0;
           break;
         }
     case 3:
         {
           Stack< char > charStack(Number_Of_Elements);
           StackFunction(charStack);
           charStack.~Stack();
           return 0;
           break;
        }
 }
   return 0;
```

Κλάσεις με μέλη άλλα αντικείμενα

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

class Time
{     //Time class
  public:
     Time();
     Time(int, int);
     void setTime(int, int);
     void getTime(int&, int&);
     void printTime();

  private:
     int hr;
     int min;
};
```

Κλάσεις με μέλη άλλα αντικείμενα (composition)

```
class Date
{
  public:
      Date();
      Date(int, int, int);
      void setDate(int, int, int);
      void getDate(int&, int&, int&);
      void printDate();
  private:
      int month;
      int day;
      int year;
};
class Event
{
  public:
    Event(int hours = 0, int minutes = 0, int d = 1,
          int m = 1, int y = 1900, string name = " ");
    void setEventData(int hours, int minutes, int d, int m,
                      int y, string name);
    void printEventData();
  private:
    string eventName;
    Time eventTime;
    Date eventDay;
};
```

Η κλάση `Enevnt` έχει μέλη αντικείμενα `Time` και `Date`

Συναρτήσεις μέλη της `Time` (1)

```
Time::Time()
{       //default constructor
        hr = 0;
        min = 0;
}

Time::Time(int hours, int minutes)//constructor με παραμέτρους
{
        hr = (0 <= hours && hours < 24)? hours : 0 ;
        min = (0 <= minutes && minutes < 60)? minutes : 0 ;
}

void Time::setTime(int hours, int minutes)
{
        hr = (0 <= hours && hours < 24)? hours : 0 ;
        min = (0 <= minutes && minutes < 60)? minutes : 0 ;
}
```

Συναρτήσεις μέλη της `Time` (2)

```
void Time::getTime(int& hours, int& minutes)
{
      hr = hours;
      min = minutes;
}

void Time::printTime()
{
      if(hr < 10)
            cout << "0";
      cout << hr << ":";
      if(min < 10)
            cout << "0";
      cout << min << endl;
}
```

Συναρτήσεις μέλη της `Date` (1)

```
int Date::checkDay( int testDay ) const
{
   static const int daysPerMonth[ 13 ] =
    { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };


// Έλεγχος αν η τιμή της testDay είναι σωστή για τον μήνα που δίνει η month
   if ( testDay > 0 && testDay <= daysPerMonth[ month - 1 ] )
     return testDay;

// Εάν η ημέρα είναι 29 Φεβρουαρίου, έλεγχος αν το έτος είναι δίσεκτο
   if ( month == 2 && testDay == 29 && ( year % 400 == 0 ||
      ( year % 4 == 0 && year % 100 != 0 ) ) )
      return testDay;

   cout << "Invalid day (" << testDay << ") set to 1." <<endl;
   return 1;
}
```

Συναρτήσεις μέλη της `Date` (2)

```
Date::Date(): month(1),day(1),year(2000)//default constructor
{  }

Date::Date(int d, int m, int y)
{//constructor με παραμέτρους
      month = (m >= 1 && m <= 12)? m : 1;
      day = checkDay(d) ; //day = (d>=1 && d <= 31)? d : 1 ;
      year = (y >= 2000 && y <= 2030)? y : 2000 ;
}

void Date::setDate(int d, int m, int y)
{//ρύθμιση ημερομηνίας (με έλεγχο)
      month = (m >= 1 && m <= 12)? m : 1;
      day = checkDay(d) ; //day = (d>=1 && d <= 31)? d : 1 ;
      year = (y >= 2000 && y <= 2030)? year : 2000 ;
}

void Date::getDate(int& d, int& m, int& y)
{
      day = d;
      month = m;
      year = y;
}
```

Συναρτήσεις μέλη της `Date` (3)

```
void Date::printDate()
{
  //εκτύπωση της ημερομηνίας (H/M/E)
    if(day < 10)
       cout << "0";
    cout << day << "/";
    if(month < 10)
       cout << "0";
    cout << month << "/";
    cout << year;
}
```

Συναρτήσεις μέλη της `Event` (1)

```
Event::Event(int hours, int minutes, int d, int m, int y, string name)
     : eventTime(hours, minutes),
                eventDay(d, m, y)
{
  eventName = name;
}
```

Πέρασμα αρχικών τιμών στις συναρτήσεις κατασκευής (constructors) των κλάσεων μελών της `Event` , `Time(eventTime)` και `Date( eventDay)` .
Κλήση του ***default copy constructor*** που δημιουργεί η C++

```
void Event::setEventData(int hours, int minutes, int d, int m, int y,
string name)
{
    eventTime.setTime(hours, minutes);
    eventDay.setDate(d, m, y);
    eventName = name;
}
```

Συναρτήσεις μέλη της `Event` (2)

```
void Event::printEventData()
{
      cout << eventName << " occurs ";
      eventDay.printDate();
      cout << " at ";
      eventTime.printTime();
      cout << endl;
}
```

```
int main()
{
      Event object;
      object.setEventData(14, 0, 14, 4, 2014, "Εξετάσεις");
   //print out the data for object
      object.printEventData();

}
```

Container classes

- Μια container class είναι ένας τύπος δεδομένων που μπορεί να περιέχει μια συλλογή στοιχείων
- Τα στοιχεία μπορεί να είναι του ίδιου τύπου (ομογενές container) ή διαφόρων τύπων (μη ομογενές)
- sequential containers: Γραμμικές δομές όπως τα διανύσματα και οι συνδεδεμένες λίστες. Στοιχεία ενός τύπου. Η αποθήκευση και η προσπέλαση γίνονται με βάση τη θέση.
- Associative containers: Μη γραμμικές δομές για γρήγορη αναζήτηση στοιχείων με βάση ένα κλειδί (key
- Στη C++ οι container classes μπορούν να υλοποιηθούν σαν μία κλάση (απλή ή σύνθετη) με συναρτήσεις μέλη (αλγόριθμους) για την επεξεργασία των στοιχείων της (προσθήκη, αφαίρεση, τροποποίηση, αναζήτηση, ταξινόμηση, πλήθος στοιχείων κλπ).
- Ενας αλγόριθμος μπορεί να είναι γενικός, ανεξάρτητος του τύπου των στοιχείων. Για την ταξινόμηση αρκεί να έχουν ορισθεί οι τελεστές σύγκρισης (> , == , <)
-

Iterators

Στους στάνταρ πίνακες (arrays) χρησιμοποιούνται δείκτες (pointers) για τη προσπέλαση και επεξεργασία των στοιχείων τους. Στα containers ο αντίστοιχος δείκτης ονομάζεται iterator και έχει παρόμοιες ιδιότητες. Είναι ένα αντικείμενο που διατρέχει τα στοιχεία ενός container

Κατηγορίες Iterators

- **input :** Χρησιμοποιείται για την ανάγνωση στοιχείων ενός container με φορά από την αρχή προς το τέλος (unidirectional).
- **output :** Χρησιμοποιείται για την εγγραφή στοιχείων σε ένα container με φορά από την αρχή προς το τέλος (unidirectional).
- **forward :** Συνδυάζει ανάγνωση και εγγραφή και διατηρεί τη θέση του στοιχείου
- **bidirectional :** Έχει τις ιδιότητες του Forward αλλά έχει τη δυνατότητα μετακίνησης από το τέλος προς την αρχή
- **random access :** Έχει τις ιδιότητες του bidirectional και επιπλέον επιτρέπει τη τυχαία προσπέλαση των στοιχείων.

Λειτουργίες iterator

- Ολοι
  `++p , p++`
- Input
  `*p , p1 = p2 , p1 == p2 , p1 != p2`
- Output
  `*p , p1 = p2`
- Farward
  Εχουν όλες τις λειτουργίες των Input και Output
- Bidirectional
  `--p , p--`
- Random-access
  `p += i , p -= i , p + i , p – i , p[i] ,`
  `p1 <= p2 , p1 >= p2 , p1 > p2`

Συναρτήσεις για iterators

- `begin()` Επιστρέφει ένα iterator στην αρχή ενός container
- `end()` Επιστρέφει ένα iterator στο τέλος ενός container
- `cbegin()` Επιστρέφει ένα σταθερό iterator (read only) στην αρχή ενός container
- `cbend()` Επιστρέφει ένα σταθερό iterator (read only) στο τέλος ενός container

–

- `container::iterator` iterator για ανάγνωση/εγγραφή
- `container::const_iterator` iterator μόνο για ανάγνωση

Παράδειγμα iterator

```
#include <iostream>
using namespace std;
#include <iterator> // ostream_iterator και istream_iterator

int main()
{
   cout << "Enter two integer numbers : ";

   istream_iterator< int > input_number( cin );

   int number1 = *input_number; // διάβασμα του πρώτου αριθμού
   ++input_number;              // μετακίνηση του iterator στον επόμενο
   int number2 = *input_number; // διάβασμα του δεύτερου αριθμού

   // δημιουργία ostream_iterator για την cout
   ostream_iterator< int > output_number( cout );

   cout << "The sum is: ";
   *output_number = number1 + number2; // output result to cout
   cout << endl;

  return 0;
}
```

- Παραδείγματα containers από τη Standard Template Library (STL) της C++
  - String :
  - Vector : Πίνακας μίας, δυναμικά μεταβλητής, διάστασης

Αρχικοποίηση container

`C<T> c;` Δημιουργία ενός κενού container με όνομα `c` και με στοιχεία τύπου `T`

```
vector<int> v1;
```

`C c(c2);` Δημιουργία ενός container `c` αντιγράφου του `c2`

```
vector<int> v1(v2);
```

`C c(n,t);` Δημιουργία ενός container `n` στοιχείων με τιμή `t`

```
vector<int> v1(n,i)
```

`C c(b,e);` Δημιουργία ενός container `n` στοιχείων με τιμή `t`

`C c(n);` Δημιουργία ενός container `n` στοιχείων με τιμή `0`

```
vector<int> v1(n);
```

String container

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

int main()
{
  string message("Hello World");
  string month = "January";
  string s1(5,'a');
  string s3;
  s3 ='a';
  //string s3 = 'a'
  //string s3('a');
  //string s4 = 11;
  //string s4(5);
  string msg1;
  cin >> msg1;
  cout << msg1;

  getline( cin, msg1 );
  cout << msg1;
```

5 χαρακτήρες , a

είσοδος : Hello World
έξοδος : Hello

είσοδος : Hello World
έξοδος : Hello World

Strings - 2

```
string msg1,msg2;
msg1 = “asdfgh”;
msg1=msg2;

msg1[3]='F'
```

Η μεταβλητή τύπου `string` είναι array

```
string msg3;
msg3 = msg1+ “jkl”;


msg1 = msg1 + “jkl”;
msg1 = msg1.append(msg2);

int str_length;
str_length = msg1.length();
msg1.resize(msg1.length() + 10);
a = msg1.capacity()

msg1.empty();
```

προσθήκη χαρακτήρων

μήκος string

προσθήκη 10 κενών θέσεων

πλήθος χαρακτήρων χωρίς τις κενές θέσεις

true / false

Strings - σύγκριση

```
int main()
{
  string str1 = “ASDFHJKL”;
  string str2 = “asdfhjkl”;
  if(str1 == str2)
    cout << "Equal strings";
  else
    if(str1 > str2)
      cout << "Str1 > str2";
    else
      cout <<"str1 < str2";
```

αποτέλεσμα

```
 int result;
 result = str1.compare(str2);
```

-1 :(str1<str2) , 0: (str1<str2) , 1:(str1<str2)

```
 result = str1.compare(2,5,str2,1,5);
}
```

Strings - substring

```
    string string_1 = “abcdefghijklmnop”;
    cout << string_1.substr(5,4);
```

αποτέλεσμα fghi

Strings - swap

```
string first = "first";
string second = "second" );
first.swap( second );          // αντιμετάθεση strings
```

Strings – αναζήτηση

```
int result;
string str1 = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
result = str1.find("GHI");          αποτέλεσμα 6
result = str1.rfind("GHI");         αναζήτηση απο το τέλος προς την αρχή

result = str1.find_first_of( str2); η πρώτη σύμπτωση ενός χαρακτήρα του
                                    str2 στο str1. Αναζήτηση από την αρχή

result = str1.find_last_of( str2);  η τελευταία σύμπτωση ενός χαρακτήρα
                                    του str2 στο str1. Αναζήτηση από το τέλος

result = str1.find_first_not_of( str2); η θέση του πρώτου χαρακτήρα του
                                        str1 που δεν υπάρχει στο str2

result = str1.find_last_not_of( str2);
```

Αν αποτύχει η αναζήτηση επιστρέφεται η τιμή -1 (`string::npos`)

Strings – αντικατάσταση

```
str1.replace( position, 1, "." );
```

αντικατάσταση στη θέση `position` ενός χαρακτήρα με `.`

```
str1.replace( position, n, str2, k, m );
```

`Position` : η θέση απο την οποία θα αρχίσει η αντικατάσταση
`n` πόσοι χαρακτήρες θα αντικατασταθούν
`str2` Οι χαρακτήρες που θα αντικαταστήσουν τους χαρακτήρες του `str1`
`k` Ο πρώτος χαρακτήρας του `str2`,
`m` το πλήθος χαρακτήρων του `str2`

```
string str1 = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
string str2 = "cdefghijk";
string1.replace( 4, 4, str2, 3, 4 );
```

"ABCDefghIJKLMNOPQRSTUVWXYZ"

Strings – διαγραφή

```
str1.erase( n );
```

Διαγραφή των χαρακτήρων από τη θέση `n` έως το τέλος του `str1`

Μετατροπές strings

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string string1 = "C++ string class";
   int len = string1.length();
   char *s_ptr1 = new char[len +1];
   string1.copy( s_ptr1, len, 0 );
   s_ptr1[ len ] = '\0';

   const char *s_ptr2 = NULL;
   s_ptr2 = string1.data(); δεν προσθέτει '\0'

   char c_string[] = string1.c_str();  προσθέτει '\0'



   delete [] s_ptr1;
   return 0;
}
```

iterators

```
#include <iostream>
#include <string>
using namespace std;
int main()
{
   string string1( "Testing iterators" );
   string::const_iterator iterator1 = string1.begin();
   cout << "string1 = " << string1
        << "\n(Using iterator iterator1) string1 is: ";

   while ( iterator1 != string1.end() )
   {
      cout << *iterator1++; // dereference iterator

   }
   cout << endl;
   return 0;
}
```

Τύπος vector

Ένας δυναμικός πίνακας μίας διάστασης του οποίου τα στοιχεία μπορεί να είναι μεταβλητές, δομές ή αντικείμενα

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

int main()
{
   vector <double> d_vector_1(10);
   vector <double> d_vector_2(10);

   d_vector_1[3] = 11.3;
   d_vector_1.at(3)
   inputVector( d_vector_1 );

}

void inputVector( vector< double > &array )
{
  for ( size_t i = 0; i < array.size(); i++ )
     cin >> array[ i ];
}
```

Τύπος vector

```
#include <iostream>
#include <vector>
#include <iterator>

using namespace std;

int main()
{
   vector <int> i_vector_1; // κενό vector
   vector <int> i_vector_2; // κενό vector
   const int v_size = 5;

   cout << "i_vector_1 size = "<<i_vector_1.size()<<endl;// 0

   for(int i=0;i<v_size;i++)
       i_vector_1.push_back(i+1);//προσήκη στοιχείων στο τέλος
                                 (1,2,3,4,5)

   cout << "i_vector_1 size = "<<i_vector_1.size()<<endl;// 5
   i_vector_2 = i_vector_1;

   for(int i=0;i<i_vector_2.size();i++)
      cout << "i_vector_2["<<i<<"] = "<<i_vector_2[i]<<endl;
                          // (1,2,3,4,5)
```

Τύπος vector

```
if (i_vector_1 == i_vector_2)
        cout << "equal arrays";
    else
        cout << "different arrays";
     cout <<endl<<endl;

    int new_size = 10;
    i_vector_1.resize(new_size); // (1,2,3,4,5,0,0,0,0,0)
    cout << "i_vector_1 size = "<<i_vector_1.size()<<endl; // 10

    if (i_vector_1 == i_vector_2)
        cout << "equal arrays";
    else
        cout << "different arrays";
    cout <<endl<<endl;

    for(int i=0;i<i_vector_1.size();i++)
       cout << "i_vector_1["<<i<<"] = "<<i_vector_1[i]<<endl;
                        // (1,2,3,4,5,0,0,0,0,0)

    i_vector_1.erase(i_vector_1.begin()+2); // (1,2,4,5,0,0,0,0,0,0)
    i_vector_1.erase(i_vector_1.begin(),i_vector_1.begin+2);
                          //    (4,5,0,0,0,0,0,0)
    i_vector_1.clear(); // κενό
```

Τύπος vector

```
cout << "i_vector_1 size = "<<i_vector_1.size()<<endl;

vector<int>::iterator iterator_1 = i_vector_2.begin();
i_vector_2.insert(iterator_1+1, 2 , 100 );

for(int i=0;i<i_vector_2.size();i++)
   cout << "i_vector_2["<<i<<"] = "<<i_vector_2[i]<<endl;
```

(1,100,100,2,3,4,5)

```
iterator_1 = i_vector_2.begin();
i_vector_2.insert(iterator_1+1, 200 );

for(int i=0;i<i_vector_2.size();i++)
   cout << "i_vector_2["<<i<<"] = "<<*(iterator_1 + i)<<endl;
```

(1,200,100,100,2,3,4,5)

```
i_vector_2.pop_back(); //
```

(1,200,100,100,2,3,4)

Τύπος vector
Πίνακες δύο ή περισσοτέρων διαστάσεων

```
#include <iostream>
#include <vector>

using namespace std;

main()
{
   vector<int> Matrix_1x2(2,0);
   vector<vector<int> > Matrix_3x2(3,Matrix_1x2);

   //   ισοδύναμος ορισμός
   // vector< vector<int> >  Matrix_3x2(3, vector<int>(2,0));

   int ii, jj;
   for(ii=0; ii < 3; ii++)
   {
      for(jj=0; jj < 2; jj++)
      {
         cout << Matrix_3x2[ii][jj] << endl;
      }
   }
}
```

```
vector<int> Matrix_1x3(3,0); // Διάνυσμα τριών στοιχείων με αρχική τιμή 0


vector< vector<int> > Matrix_4x3(4, Matrix_1x3);
// Πίνακας 4x3 : Διάνυσμα με στοιχεία 4 διανύσματα


vector< vector< vector<int> > > Matrix_5x4x3(5, Matrix_4x3);
// Πίνακας 5x4x3



 vector< vector< vector<int> > >
     Matrix_(5, vector< vector<int> > (3, vector<int>(4,0))
```

Vector με στοιχεία string

```
#include <iostream>
#include <string>
#include <vector>
#include <iterator>

int main()

{
   vector<string> string_array(3);
   string_array[0] = "String 1";
   string_array[1] = "String 2";
   string_array[2] = "String 3";

   string_array.push_back("string 4");

   vector<string>::iterator iterator_1 = string_array.begin();
   string_array.insert(iterator_1+2,"New String");
   for(int i=0;i<string_array.size();i++)
      cout << "string_array["<<i<<"] = "<<*(iterator_1 + i)<<endl;
   return 0;
 }
```

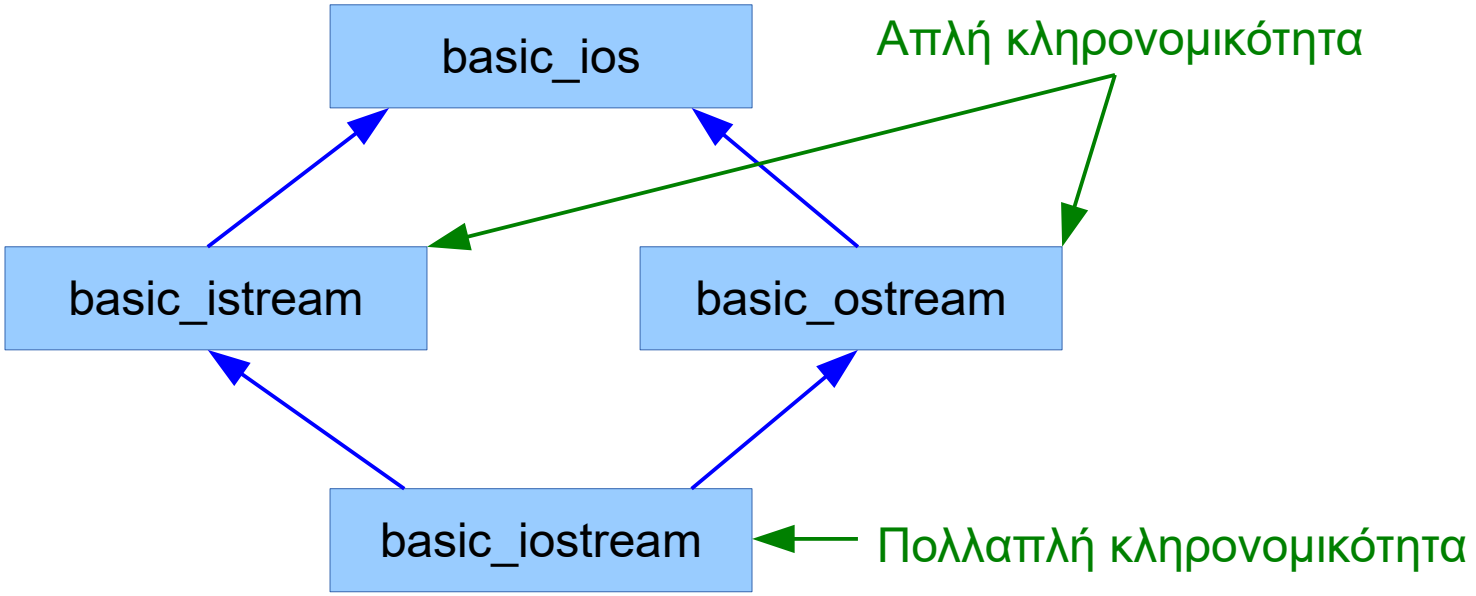
String 1
String 2
New
String
String 3
string 4

Είσοδος – έξοδος I/O

- Οι λειτουργίες εισόδου-εξόδου περιέχονται στη στάνταρ βιβλιοθήκη της C++ (υλοποιούνται με ιεραρχίες κλάσεων)
- Οι συναρτήσεις μέλη που υποστηρίζουν ένα τύπο στοιχείων δεν μπορούν να χρησιμοποιηθούν με άλλο τύπο (type-safe I/O)
- Η είσοδος και η έξοδος βασίζονται σε streams (ρεύμα, ροή)
  - **Stream**: Μία ακολουθία bytes που χρησιμεύει σαν μια ενδιάμεση μνήμη (buffer) και περιέχει τα προς επεξεργασία δεδομένα.
  - **Input sream** : Πληκτρολόγιο , αρχείο , δίκτυο.
  - **Output stream** : Οθόνη, αρχείο, εκτυπωτής
  - Εξασφαλίζεται ή ομαλή συνεργασία αργών περιφερειακών (keyboard, mouse, printer) με την κεντρική μονάδα.
- **Low-Level I/O (Unformated I/O)**: Ενας αριθμός bytes (raw data) μεταφέρονται από τη μνήμη σε μία συσκευή ή αντίστροφα. Για υψηλή ταχύτητα μεταφοράς και μεγάλο όγκο δεδομένων (λειτουργικό σύστημα,DMA)
- **High-Level I/O (Formated I/O)**: Τα bytes ομαδοποιούνται ώστε να παριστάνουν συγκεκριμένους τύπους στοιχείων (ακέραιοι , πραγματικοί, χαρακτήρες, strings). Η μορφή αυτή είναι πιο "*φιλική*" προς τον προγραμματιστή.

iostream

- **istream library**: Ο τελεστής **>>** αφαιρεί τιμές από το ρεύμα εισόδου και τις αποδίδει σε μεταβλητές ή αντικείμενα.
- **ostream library:** Ο τελεστής **<<** προσθέτει τιμές στο ρεύμα εξόδου
- **iostream library** (iostream headers): Παρέχει όλες τις βασικές λειτουργίες εισόδου-εξόδου. Περιέχει πολλά class templates (basic_istream, basic_ostream, basic_iostream)



Ιεραρχία class templates

Standard Stream Objects

- **cin** : στιγμιότυπο (instance) της istream. *Συνδέεται* με τη στάνταρ συσκευή εισόδου (πληκτρολόγιο).
  - `cin >> a .` Ο τελεστής δείχνει τη φορά ροής των δεδομένων (απο τη συσκευή πρός τη μνήμη)
- **cout** : στιγμιότυπο (instance) της ostream. *Συνδέεται* με τη στάνταρ συσκευή εξόδου (οθόνη).
  - `cout << a .` Ο τελεστής δείχνει τη φορά ροής των δεδομένων (απο τη μνήμη προς τη συσκευή)
- **cerr** : στιγμιότυπο (instance) της ostream. *Συνδέεται* με τη στάνταρ συσκευή σφάλματος. Κάθε εισαγωγή στο ρεύμα εμφανίζεται αμέσως στην έξοδο (unbuffered)
- **clog** : στιγμιότυπο (instance) της ostream. *Συνδέεται* με τη στάνταρ συσκευή σφάλματος. Εμφάνιση στην έξοδο μόνο όταν γεμίσει ο buffer (buffered)

ostream
συνάρτηση `put`

```
cout.put('A')
cout.put(65).
cout.put('\n')
```

έξοδος ενός χαρακτήρα

αλλαγή γραμμής

istream - Συναρτήσεις μέλη

`get()` Διαβάζει ένα χαρακτήρα από το ρεύμα εισόδου
`eof()` επιστρέφει true αν έχει φτάσει το τέλος του αρχείου, αλλιώς επιστρέφει false
`getline()` Διαβάζει μία γραμμή, μαζί με τα κενά

```
#include <iostream>
using namespace std;
int main()
{ int ch; // int γιατί ο τύπος char δεν είνα συμβατός με το EOF

  cout << "Before input, cin.eof() is " << cin.eof() << endl
       << "Enter a sentence followed by end-of-file:" << endl;
  // get για διάβασμα χαρακτήρων : put για έξοδο
  while ( ( ch = cin.get() ) != EOF )
      cout.put(ch);

  cout << "\nEOF in this system is: " << ch << endl;
  cout << "After input of EOF, cin.eof() is " << cin.eof() << endl;
  return 0;
}
```

Συναρτήσεις get, getline και gcount

```
int main()
{
   const int buffer_size = 100;
   char buffer1[buffer_size],buffer2[buffer_size];
   cout << "Enter a string with spaces : ";
   cin >> buffer1;  // διαβάζει μέχρι το πρώτο κενό
   cout << "String entered : "<< buffer1 << endl<<endl;
   cout << "Number of characters read  : "<<cin.gcount<<endl;
   cin.get(buffer2,buffer_size);//διαβάζει μέχρι το τέλος της γραμμής
   cout << "Number of characters read  : "<<cin.gcount<<endl;
   cout << "String entered : "<< buffer2 << endl<<endl;
   return 0;
}
```

Είσοδος : aaaaa bbbbbb ccccc ddddddd

Πρώτη έξοδος : aaaaa
Δεύτερη έξοδος : bbbbbb ccccc ddddddd

`cin.getline( buffer, buffer_size);`

Όμοια συμπεριφορά με την `cin.get(buffer,buffer_size)` αλλά διαβάζει και το χαρακτήρα `\n` (αλλαγή γραμμής)

`cin.gcount();`

επιστρέφει τον αριθμό των χαρακτήρων που διάβαστηκαν με την `cin`

Συναρτήσεις `read`, `write` και `gcount`

```
#include <iostream>
using namespace std;
int main()
{
   const int buffer_size = 100;
   char buffer[buffer_size];

   cout << "Enter a string : ";
   cin.read(buffer,20);
   cout <<endl<< "String is : " << endl;
   cout.write(buffer, cin.gcount());
   cout << endl;
  return 0;
}
```

- ignore() απορρίπτει το πρώτο χαρακτήρα
- ignore(n) απορρίπτει τους n πρώτους χαρακτήρες.
- peek() διαβάξει ένα χαρακτήρα από το ρεύμα χωρίς να τον αφαιρεί από το ρεύμα .
- unget() επιστρέφει στο ρεύμα τον τελευταίο χαρακτήρα που διαβάστηκε ώστε να μπορεί να διαβαστεί την επόμενη φορά.
- putback(char ch) τοποθετεί ένα χαρακτήρα στο ρεύμα .

Stream Manipulators

Καθορίζουν την μορφή των δεδομένων στην έξοδο (formated output)

Σύστημα αρίθμησης
`dec` (δεκαδικό) `hex` (δεκαεξαδικό) `oct` (οκταδικό)
`setbase(n)  n = 10,8,16  ( #include <iomanip> )`

```
cout << a << hex << c << d << dec << c;
```

πραγματικοί

`precision  setprecision :` καθορίζουν το πλήθος ψηφίων

```
double x = 12.3456789;
cout.precision(2);
cout<< x << endl;                          12
cout.precision(4);
cout << x << endl;                         12.34
cout << setprecision(8) << x <<endl;       12.345679

cout << sohowpoint << endl;
cout << precision(2)<< x ;     12.
```

`fixed , scieentific :`

```
cout << x<<endl;                  12.3457
cout<<scientific<<x<<endl;        1.234568e+001
cout <<fixed<<x<< endl;           12.345679

cout <<setprecision(9)<<fixed<<x<< endl; 12.34567900
```

Έξοδος πραγματικών

```
int defaultPrecision = cout.precision();
cout << setiosflags(ios_base::fixed);
cout.precision(2);
cout.precision( defaultPrecision );
cout << resetiosflags(ios_base::fixed);
cout << setiosflags(ios_base::scientific);
cout << resetiosflags(ios_base::fixed);
```

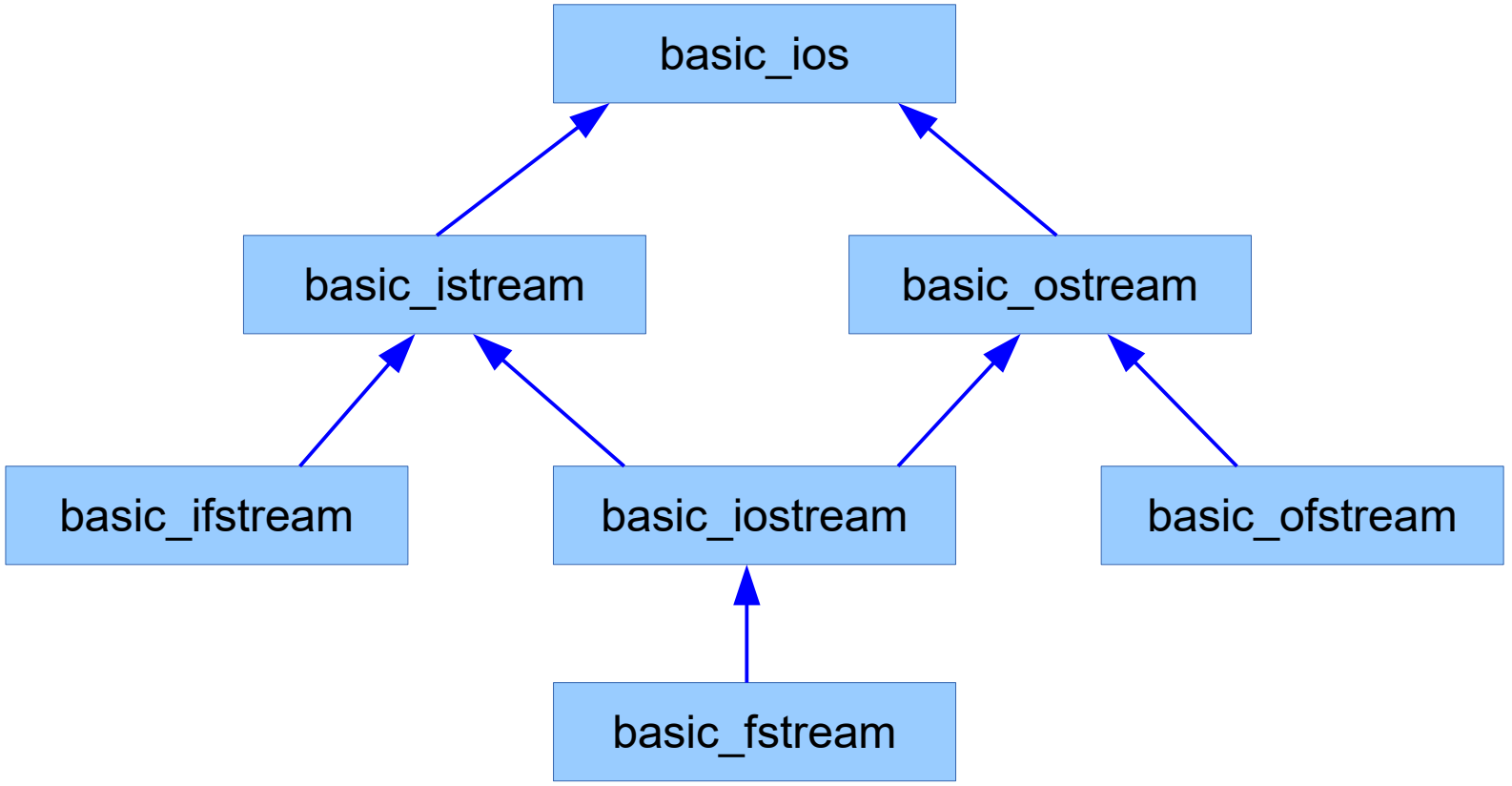
Εύρος πεδίου εκτύπωσης (`width, setw`)

```
#include <iostream>
#include <stdio.h>
#include <iomanip>
using namespace std;
int main()
{
   char ch1[20];
   cin.width(5);
   cin >> ch1;
   cout.width(10);
   cout << ch1 << endl;
   cout << setw(10) << ch1 << endl;

   return 0;
}
```

Είσοδος `ABCDEFGH` (διαβάζονται οι 5 πρώτοι χαρακτήρες)
Έξοδος `  ABCDE` (5 κενά, 5 χαρακτήρες)

Αρχεία

- Όταν ανοίγει ένα αρχείο δημιουργείται ένα αντικείμενο (στιγμιότυπο) και ένα ρεύμα (stream) συνδέεται με αυτό. Το ρεύμα αποτελεί ένα κανάλι επικοινωνίας ενός προγράμματος με ένα αρχείο ή μία συσκευή
- Για την επεξεργασία αρχείων απαιτούνται τα αρχεία επικεφαλίδων `<iostream>` και `<fstream>` για τα πρότυπα κλάσεων `basic_ifstream` (είσοδος) , `basic_ofstream` (έξοδος) `basic_fstream` (είσοδος/έξοδος)



Sequential File

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
   ofstream out_data_file("data_file.txt",ios::out);

   if(!out_data_file)
   {
      cout << "Error opening file"<<endl;
      return 1;
   }
   out_data_file << "This is  a Sequential file";
   out_data_file.close()//
   char string1[30];
   ifstream in_data_file( "data_file.txt", ios::in );
   if(!in_data_file)//
   {
      cout << "Error opening file"<<endl;
      return 1;
   }
   in_data_file >> string1;
   return 0;//
}
```

Όνομα αρχείου

Τρόπος Προσπέλασης

έλεγχος αν είναι δυνατό να ανοιχθεί το αρχείο

κλείνει το αρχείο

Κενό : είσοδος

έλεγχος αν είναι δυνατό να ανοιχθεί το αρχείο

η συνάρτηση καταστροφής της κλάσης κλείνει το αρχείο

`out_data_file` : επιστρέφει `true` αν το αρχείο μπορεί να ανοιχθεί
αλλιώς επιστρέφει `false`

Περιπτώσεις που επιστρέφεται `false` :

- Απόπειρα να ανοιχθεί για ανάγνωση ένα αρχείο που δέν υπάρχει

Τρόποι ανοίγματος αρχείων

`ios::app`	Προσθήκη στο τέλος του αρχείου
`ios::ate`	Άνοιγμα αρχείου για ανάγνωση/εγγραφή και μετακίνηση στο τέλος του.
`ios::in`	Άνοιγμα αρχείου για είσοδο.
`ios::out`	Άνοιγμα αρχείου για έξοδο. Αν το αρχεί υπάρχει τα περιεχόμενα του διαγράφονται. Αν δεν υπάρχει δημιουργείται νέο.
`ios::trunc`	Άν το αρχείο υπάρχει διαγράφεται.
`ios::binary`	Άνοιγμα αρχείου για ανάγνωση/εγγραφή σε δυαδική μορφή (όχι κείμενο).
`ios::nocreate`	Άνοιγμα αρχείου μόνο αν υπάρχει.
`ios::noreplace`	Άνοιγμα αρχείου μόνο αν δεν υπάρχει.

seek

```
#include <iostream>
#include <fstream>
#include <iomanip>
using namespace std;
int main()
{
   ofstream out_data_file("data_file.txt",ios::out);

   if(!out_data_file)// έλεγχος αν είναι δυνατό να ανοιχθεί το αρχείο
   {
      cout << "Error opening file"<<endl;
      return 1;
   }
   for (int i =0;i<20;i++)
       out_data_file <<setw(5)<<left<< "Line"
                     <<setw(5)<<right << i+1;

   out_data_file.seekp(ios::beg);
   out_data_file.seekp(33);
   //out_data_file << 0<<endl;
   //out_data_file <<'\n'<<setw(5)<<left<< "Line"
                   <<setw(5)<<right<< "new"<<endl;
   out_data_file.close();//κλείνει το αρχείο
```

```
 ifstream in_data_file("data_file.txt",ios::in);
 string string_1;
 int k;

if(!in_data_file)// έλεγχος αν είναι δυνατό να ανοιχθεί το αρχείο
 {
    cout << "Error opening file"<<endl;
    return 1;
 }

 in_data_file.seekg(ios::beg);
 in_data_file.seekg(40,ios::cur);
 in_data_file >> string_1 >> k;

 cout << string_1<<setw(5)<<k<<endl;
 return 0;
```

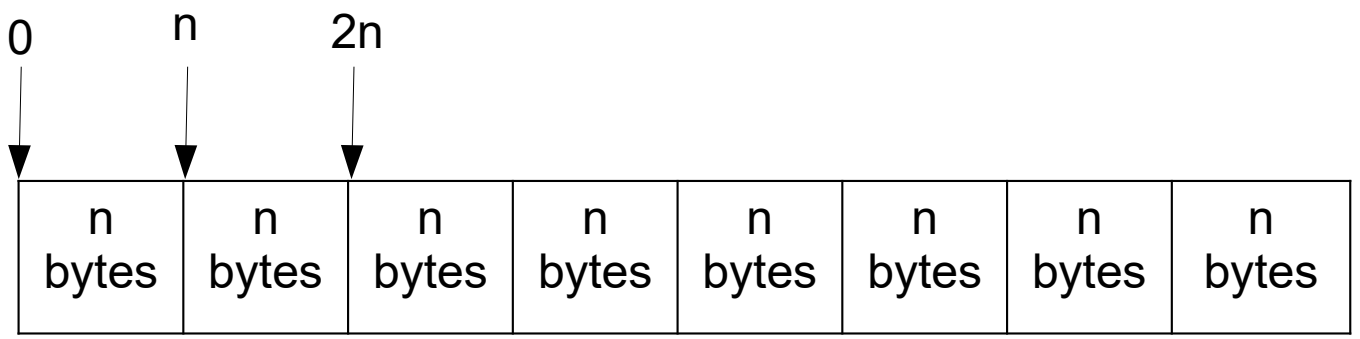
Συνάρτηση `seekg (seek get)` για διάβασμα από αρχείο
Συνάρτηση `seekp (seek put)` για γράψιμο σε αρχείο

```
file.seekg(ios::beg);     στην αρχή
file.seekg(ios::end);     στο τέλος
file.seekg(n);            στο byte n (από την αρχή)
file.seekg(n,ios::beg);   n bytes από την αρχή
file.seekg(n,ios::cur);   n bytes από τη παρούσα θέση
file.seekg(n,ios::end);   n bytes από το τέλος
```

```
location = file.tellg();επιστρέφει τη παρούσα θέση
```

Random-Access Files

- Η ενημέρωση σειριακών (ακολουθιακών) αρχείων μπορεί να προκαλέσει σε αλλοίωση του περιεχομένου τους.
- Η C++ δεν επιβάλλει συγκεκριμένη οργάνωση αρχείων
- Απλή μορφή αρχείου τυχαίας προσπέλασης. Εγγραφές σταθερού μήκους



- Δεδομένα μπορούν να γραφούν σε μία θέση χωρίς επηρεάζονται οι άλλες.

Δυαδικά αρχεία

- Στη C++ τα αρχεία ανοίγονται ως αρχεία κειμένου, αν δεν οριστεί διαφορετικά.
- Στα αρχεία κειμένου γίνονται ορισμένες μεταφράσεις χαρακτήρων (π.χ. Carriage Return, Line Feed sequences CR + LF σε αλλαγή γραμμής).
- Αριθμητικά στοιχεία. Σε μορφή κειμένου ο αριθμός 10000 απαιτεί 5 bytes. Σε δυαδική μορφή απαιτεί 2 bytes.
- Στα δυαδικά δεν υπάρχει ο χαρακτήρας EOF

```
istream &read(char *buf, streamsize num);
ostream &write(const char *buf, int streamsize num);
```

`*buf` δείκτης σε μια περιοχή μνήμης (buffer)
`num` αριθμός των bytes για εγγραφή ή ανάγνωση

```
istream &get(char &ch);
ostream &put(char ch);
```

Γράψιμο και διάβασμα ομάδων (Blocks) δεδομένων

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

int main()
{
    const int array_size = 7;
    int n[array_size] = {1, 2, 3, 4, 5, 6, 7 };
    int i;

    ofstream out_file("test.dat", ios::out | ios::binary);

    if(!out_file)
    {
        cout << "Cannot open file.\n";
        return 1;
    }

    out_file.write(reinterpret_cast< const char * >(n),
                   sizeof(n));
    //in_file.read((char *) n, sizeof n);
    out_file.close();
```

Πράξη ή bit προς bit

Ένας δείκτης σε χαρακτήρα δείχνει σε ένα πίνακα ακεραίων

μετατροπή όπως στη C

Γράψιμο και διάβασμα ομάδων (Blocks) δεδομένων

```
    for(i=0; i<array_size; i++)// μηδενισμός των στοιχείων του πίνακα
        n[i] = 0;

    ifstream in_file("test.dat", ios::in | ios::binary);
    if(!in_file)
    {
        cout << "Cannot open file.\n";
        return 1;
    }
    in_file.write(reinterpret_cast< const char * >(n),
          sizeof(n));

    for(i=0; i<array_size; i++) //τιμές από το αρχείο
        cout << n[i] << " ";

    in_file.close();

    return 0;
}
```

Γράψιμο και διάβασμα ανά ένα byte δεδομένων

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
  char *p_string = "This is a string";
  ofstream out_file("test1.dat", ios::out | ios::binary);
  if(!out_file)
  {
    cout << "Cannot open file." << endl;
    return 1;
  }
  while(*p_string)
    out_file.put(*p_string++);
  out_file.close();
  char ch;
  ifstream in_file("test1.dat", ios::in | ios::binary);
  if(!in_file)
  {
    cout << "Cannot open file." << endl;
    return 1;
  }
  while(in_file.get(ch))
    cout << ch;
  in_file.close();
```

Σύγκριση αρχείων

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

int main(int argc, char *argv[])
{
  int i;
  const int buffer_size = 1024;
  unsigned char buf1[buffer_size], buf2[buffer_size];

  if(argc!=3)
  {
    cout << "Usage: compfiles <file1> <file2>" << endl;
    return 1;
  }
  ifstream file1(argv[1], ios::in | ios::binary);
  if(!file1)
  {
    cout << "Cannot open first file." << endl;
    return 1;
  }
```

```
ifstream file2(argv[2], ios::in | ios::binary);
if(!file2)
{
  cout << "Cannot open second file." << endl;
  return 1;
}

cout << "Comparing files... " << endl;
long long block_number = 0;
do
{
  //διάβασμα από τα αρχεία, έως 1024 Bytes κάθε φορά
  file1.read(reinterpret_cast<const char *>(buf1), sizeof buf1);
  file2.read(reinterpret_cast<const char *>(buf2), sizeof buf2);

  if(file1.gcount() != file2.gcount())
  {
    // αν από τα αρχεία διαβάστηκαν διαφορετικοί αριθμοί bytes
    cout << "Files have differing sizes." << endl;
    file1.close();
    file2.close();
    return 0;
  }
```

```
// αν τα δύο buffers έχουν το ίδιο πλήθος στοιχείων, σύγκριση ένα πρός ένα
    for(i=0; i<file1.gcount(); i++)
      if(buf1[i] != buf2[i])
      {
         cout << "Files are different." << endl;
         cout << "First different byte at position : ";
         cout << block_number * buffer_size + i + 1 << endl;
         file1.close();
         file2.close();
         return 0;
      }
    block_number++;
  } while(!file1.eof() && !file2.eof());// όσο δεν έχει βρεθεί το τέλος του
                                        // ενός τουλάχιστον αρχείου

  cout << "Files are the same." << endl;
  file1.close();
  file2.close();
  return 0;
}
```

Input/Output of Objects