

# Introducere în programare

## 2016 - 2017

### 2

Bogdan Pătruț

[bogdan@info.uaic.ro](mailto:bogdan@info.uaic.ro)

după Corina Forăscu

<http://profs.info.uaic.ro/~introp>

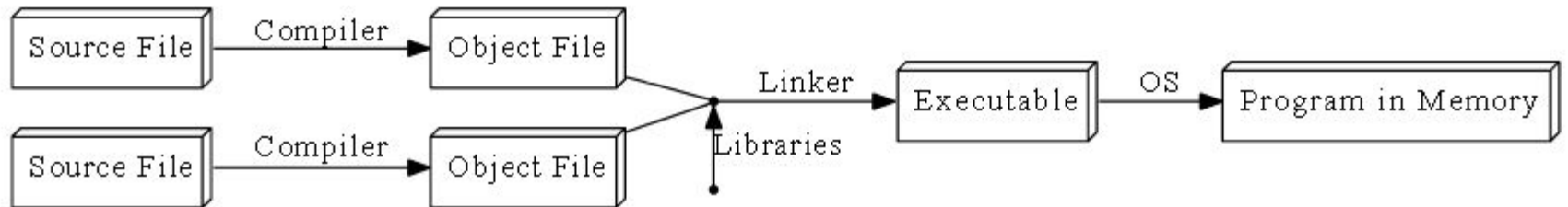
# Curs 2: conținut

- Tipuri fundamentale de date
- Caractere, cuvinte rezervate
- Variabile, expresii, asignări
- Operatori
- Expresii booleene
- Constante

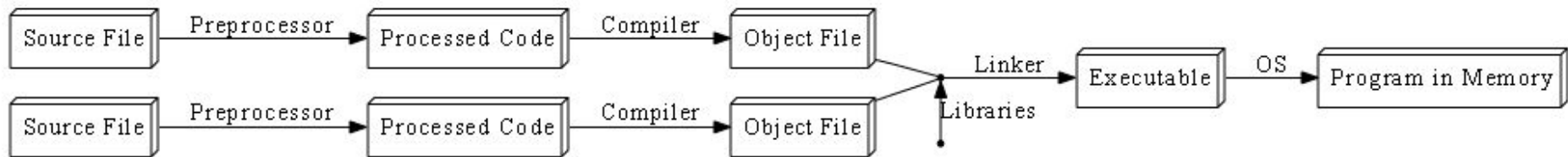
# Programare

- De ce C++
  - Puternic, flexibil (conversii)
  - Structurat
  - Portabilitate (Windows, Apple, Linux, UNIX)
  - Eficient, elegant
  - Perspective în alte limbaje de programare
  - Limbaj al programatorului
- De ce Code::Blocks ?
  - sunteți obișnuit cu el din liceu
  - gratuit
  - editor simplu și sugestiv
  - permite aranjarea automată a codului (Format use AStyle)

# Compilare



- În C++:



[http://en.cppreference.com/w/cpp/language/translation\\_phases](http://en.cppreference.com/w/cpp/language/translation_phases)

# Primul program (1)

```
/*  
 * primul program in C++  
 */
```

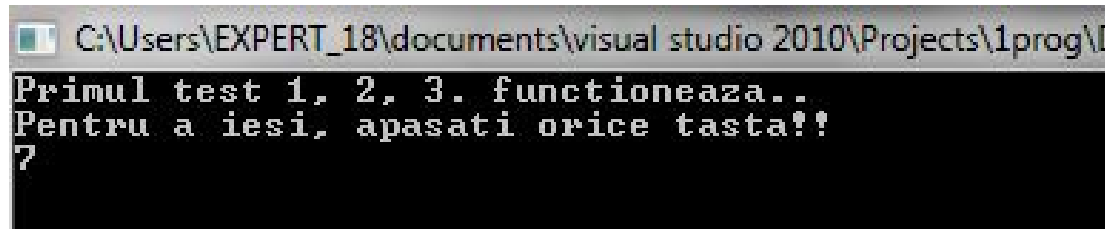
```
1.  #include <iostream>  
  
2.  int main ()  
3.  {  
4.      std::cout << "Primul test 1, 2, 3. ";  
5.      std::cout << "functioneaza.. \n";  
6.      return 0;  
7.  }
```

# Primul program (1-ieşire)

**Primul test 1, 2, 3. functioneaza..**

# Primul program

```
/*  
 * first program in C++  
 */  
  
1.  #include <iostream>  
2.  using namespace std;  
  
3.  int main ()  
4.  {  
5.  cout << "Primul test 1, 2, 3. ";  
6.  cout << "functioneaza.. \n";  
7.  char c;  
8.  cout << "Pentru a iesi, apasati orice tasta!!\n";  
9.  cin >> c;  
10. return 0;  
11. }
```



C:\Users\EXPERT\_18\documents\visual studio 2010\Projects\1prog\l  
Primul test 1, 2, 3. functioneaza..  
Pentru a iesi, apasati orice tasta!!  
?

# Forma unui program C++

```
/*comentariu; el nu influențează programul */
```

```
//directive preprocesare
```

```
#include <biblioteci> (Input/output, math, strings, ...)
```

```
#define
```

```
//declarații ale variabilelor
```

```
tipuri utilizator;
```

```
Variabile;
```

```
Funcții;
```

```
....
```

```
//funcția principală
```

```
    int main()
```

```
    {
```

```
    }
```

```
// definirea funcțiilor utilizator
```

```
.....
```

```
return 1;}      //aici se încheie programul
```



# Directiva preprocessare

## #include

### Sintaxa:

#include <*biblioteca*> (Input/output, math, strings, ...)

### Exemple:

```
#include <iostream>
```

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

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

```
#include "biblioteca_mea.h"
```

```
#include "t1.h"
```

# Directiva preprocesare #define

## Sintaxa:

#define simbol

#define simbol valoare

## Exemple:

#define infinit 1000

#define pi 3.1415926

#define then

#define si &&

#define daca if

#define altfel else

#define max(x,y) x > y ? x : y

#define min(x,y) x < y ? x : y

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

```
void main()
```

```
{
```

```
    float a,b,c;
```

```
    printf("Dati lungimile laturilor:\n");
```

```
    scanf("%f %f %f",&a,&b,&c);
```

```
    daca ((a>=0) si (b>=0) si (c>=0) si (a<b+c) si  
        (b<a+c) si (c<a+b))
```

```
        printf("%4.2f, %4.2f si %4.2f formeaza  
triunghi.",
```

```
    a,b,c);
```

```
    altfel
```

```
        printf("Nu formeaza triunghi.");
```

```
}
```

# Elementele fundamentale C++

- **Expresii** – formate din
  - **Date** reprezentate prin caracterizate de
    - Variabile
    - Constante
    - Tip
    - Nume
    - Valoare
    - Domeniu de vizibilitate
    - Timp de viață
  - **Operatori**
  - **Apeluri de funcții**

# Variabilă

```
int patrat(int n)
{ int x; x=y*y; return x; }
cout<<patrat(2);
```

- Tip = int
- Nume = x
- Valoare = 4
- Domeniu de vizibilitate: funcția *patrat*
- Timp de viață = pe durata unui apel al funcției

# Variabile globale, locale, parametri

```
#include <iostream>
```

```
int x=10,t;
```

variabile globale, vizibile și în funcția **egale**

```
char egale(int x, int y)
```

```
{ int a=1; t=x+y+a;
```

```
return x==y?'D':'N'; }
```

variabilă  
locală

parametri formali ai  
funcției *egale*, vizibili doar  
în funcția *egale*

```
void main() {
```

```
int y=2,z=3; t=1;
```

```
cout<<t<<x<<y<<z;
```

variabile globale invizibile în funcția *egale*

```
if (egale(y,z))== 'D' cout<<"Da"; else cout<<"NU";
```

```
cout<<t;
```

```
} // 1,10,2,3,Nu (deoarece y->x, adica 2->x, z->y, adica z->y și  
2!=3, apoi t=6
```

# Tip de date

- Domeniul tipului (colecția de obiecte) – **mulțime de valori** pentru care s-a adoptat un anumit **mod de reprezentare** în memorie
- Operațiile tipului
- Categoriile de tipuri de date:
  - Tipuri de date standard
  - Tipuri de date structurate de nivel jos
    - Operațiile la nivel de componentă
  - Tipuri de date de nivel înalt
    - Operațiile implementate de algoritmi utilizator

# Tipuri de date standard

- Tipuri caracter: tipurile **char**, **signed char**, **unsigned char**
- Tipuri întregi: tipurile caracter, întregi cu semn, întregi fără semn, tipurile enumerare
- Tipuri reale: tipurile întregi și tipurile flotante reale
- Tipuri aritmetice: tipurile întregi și cele flotante
- Tipuri de bază: caracter, întregi cu și fără semn, flotante
- Tipul void: desemnează o mulțime vidă

# Tipuri de date

NUME TIP	DIMENSIUNE ÎN BITI	DOMENIU
unsigned char	8	0..255
char	8	-128..127
signed char	8	-128..127
unsigned int	16	0..65535
short int, signed int	16	-32768..32767
int	16	-32768..32767
unsigned long	32	0..4.294.967.295
long, (signed) long int	32	-2.147.483.648..2.147.483.647
float	32	Şase zecimale exacte
double	64	Zece zecimale exacte
long double	80	Zece zecimale exacte



# Echivalențe între tipurile de date

`signed short int`       $\equiv$    `short`

`unsigned short int`  $\equiv$    `unsigned short`

`signed int`             $\equiv$    `int`

`unsigned int`          $\equiv$    `unsigned`

`signed long int`        $\equiv$    `long`

`unsigned long int`     $\equiv$    `unsigned long`

# Tipuri de date derivate

- Se construiesc din obiecte, funcții și tipuri incomplete:
  - tipul *tablou* de T (elementele de tip T)
  - tipul *structură*
  - tipul *uniune*
  - tipul *funcție*, derivat din tipul returnat T și numărul și tipurile parametrilor (funcție ce returnează T)
  - Tipul *pointer*, derivat dintr-un tip referențiat (tip funcție, tip obiect, tip incomplet). Valorile tipului pointer sunt referințe la o entitate de tipul referențiat (pointer la T)
- *Tipuri scalare*: tipurile aritmetice și tipul pointer
- *Tipuri agregat*: tablouri și structuri
- Un tablou de dimensiune necunoscută, o structură sau uniune cu conținut necunoscut sunt tipuri incomplete

# Caractere

- **Litere:**

A B C D ... X Y Z  
a b c d ... x y z

- **Cifre:**

0 1 2 3 4 5 6 7 8 9

- **Alte caractere:**

+ - \* / ^ \ ( ) [ ] { } = != <>  
' " \$ , ; : % ! & ? \_ # <= >= @

- **Caractere spațiu:** backspace, horizontal tab, vertical tab, form feed, carriage return

# Convenții lexicale

- Tokeni:
  - Cuvinte cheie (*Keywords*)
  - Identificatori : **bun**, **\_bun**, **bun1** VS . **rău**, **1rau**, **rau!**
    - **ALL\_CAPS** pentru constante
    - **lowerToUpper** pentru variabile, cu nume cât mai sugestive
  - Literal (constante): întregi, flotați, logici, caracter, șir caractere, secvențe escape
  - Semne de punctuație și separatori: ! % ^ & \* ( ) - + = { } | ~ [ ] \ ; ' : " < > ? , . / #
  - Operatori: aritmetici, logici, in-/decrementare, atribuire, relaționali, logici pe biți, dimensiune, condiționali, logici, de conversie
- Comentarii: /\* ...mai multe rânduri\*/  
//...pe același rând

# Cuvinte cheie (rezervate) C++

auto    const    double    float    int    short    struct    unsigned  
break    continue    elsefor    long    signed    switch    void  
case    default    enum    goto    register    sizeof    typedef    volatile  
char    do    extern    if    return    static    union    while



asm    dynamic\_cast    namespace    reinterpret\_cast    try  
bool    explicit    new    static\_cast    typeid  
catch    false    operator    template    typename  
class    friend    private    this    using  
const\_cast    inline    public    throw    virtual  
delete    mutable    protected    true    wchar\_t



and    bitand    compl    not\_eq    or\_eq    xor\_eq  
and\_eq    bitor    not    or    xor

<http://en.cppreference.com/w/cpp/keyword>

# Caractere speciale

Secvența Escape	Caracter
<code>\b</code>	Backspace
<code>\t</code>	Tab orizontal
<code>\v</code>	Tab vertical
<code>\n</code>	Linie nouă
<code>\f</code>	Pagina nouă – formfeed
<code>\r</code>	Început de rând
<code>\”</code>	Ghilimele
<code>\’</code>	Apostrof
<code>\\</code>	Backslash
<code>\?</code>	Semnul întrebării
<code>\a</code>	Alarmă

# Variabile

- Forma unei declarații:

*tip variabila;*

*tip var1, var2, ..., varn;*

*tip variabila = expresie\_constanta;*

- *Variabile globale*: declararea lor se face la începutul programului, în afara oricărei funcții.
- *Variabile locale*: declararea se face în corpul funcției, la început.

```
char c;;  
signed char sc;;
```

```
int a,b;;  
a = b = 5;;
```

```
int i;  
int suma = 0;;  
long j;
```

```
float x1, x2, x3;;  
float pi = 3.14;;  
double y;;
```

# Atribuire (Asignare)

- La execuție:
  - Lvalues (left-side) & Rvalues (right-side)
    - Lvalues variabile
    - Rvalues expresie
    - exemplu:  
distance = rate \* time;  
Lvalue: "distance,,  
Rvalue: "rate \* time"
  - Contraexemplu pt Lvalue: 5+5, "str", const int i
  - Compatibilitate
  - Conversii



# Constante întregi

- Octale: au prefixul 0 (zero)  
 $032 = 26$                        $077 = 63$
- Hexazecimale: au prefixul 0x sau 0X  
 $0x32 = 50$                        $0x3F = 63$
- Întregi “long”: au sufixul l sau L  
 $2147483647L$                        $0xaf9Fl = 44959$
- Întregi “unsigned” au sufixul u sau U  
 $345u$     $0xffffu = 65535$
- Caractere între apostrof: ‘A’, ‘+’, ‘n’
- Caractere în zecimal: 65, 42
- Caractere în octal: ‘\101’, ‘\52’
- Caractere în hexazecimal: ‘\x41’, ‘\x2A’
- Caractere speciale – secvențe escape

# Operații și funcții pentru tipurile întregi

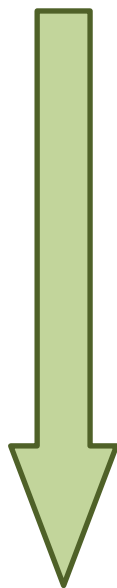
- Operații pentru tipurile întregi:

+   -   \*   /   %   ==   !=   <   <=   >  
>=   ++   --

- Funcții:

- cele de la tipul flotant
- cele din biblioteca **<ctype.h>**: **tolower, toupper, isalpha, isalnum, iscntrl, isdigit, isxdigit, islower, isupper, isgraph, isprint, ispunct, isspace**

# Operatori



Clasa de operatori	Operatori	Asociativitate
Unari	- (unar) ++ --	de la dreapta la stânga
Multiplicativi	* / %	de la stânga la dreapta
Aditivi	+ -	de la stânga la dreapta
Atribuire	=	de la dreapta la stânga
Relationali	< <= > >=	de la stânga la dreapta
De egalitate	== !=	de la stânga la dreapta
Atribuire si aritmetici binari	= *= /= %= += -=	de la dreapta la stânga

# Operatorii ++ și --

- Se aplică doar unei expresii ce desemnează un obiect din memorie (L-value):

Expresie:	<b>++i</b>	<b>i++</b>	<b>--i</b>	<b>i--</b>
Valoare	<b>i+1</b>	<b>i</b>	<b>i-1</b>	<b>i</b>
i după evaluare	<b>i+1</b>	<b>i+1</b>	<b>i-1</b>	<b>i-1</b>

**++5    --(k+1)    ++i++    nu au sens**  
**(++i)++    ok**

# Tipul flotant (real)

- **float**

- Numere reale în simplă precizie
- **sizeof(float) = 4**
- $10^{-37} \leq \text{abs}(f) \leq 10^{38}$
- 6 cifre semnificative

- **double**

- Numere reale în dublă precizie
- **sizeof(double) = 8**
- $10^{-307} \leq \text{abs}(f) \leq 10^{308}$
- 15 cifre semnificative

# Tipul flotant (real)

- **long double**

- Numere reale în “extra” dublă precizie

- **sizeof(long double) = 12**

- $10^{-4931} \leq \text{abs}(f) \leq 10^{4932}$

- 18 cifre semnificative

- Limitele se găsesc în **<float.h>**

- Operații: **+ - \* / == != < <= > >=**

# Constante reale

- Constantele reale sunt implicit **double**

125.435 1.12E2 123E-2 .45e+6 13. .56

$$1.12E2 = 1.12 \times 10^2 = 112$$

$$123E-2 = 1.12 \times 10^{-2} = 1.23$$

$$.45e+6 = 0.45 \times 10^6 = 450000$$

$$13. = 13.00 \text{ și } .56 = 0.56$$

- Pentru a fi **float** trebuie sa aiba sufixul **f** sau **F**

.56f 23e4f 45.54E-1F

$$23e4f = 23 \times 10^4 = 230000.00$$

- Pentru **long double** trebuie sa aibă sufixul **l** sau **L**

123.456e78L

# Funcții

(în bibliotecă <math.h>)

**sin cos tan**

**asin acos atan**

**sinh cosh tanh**

**exp log log10**

**pow sqrt ceil**

**floor fabs**

**ldexp frexp**

**modf fmod**



# Date booleene (logice)

- Tipul **bool**
- Domeniul de valori: {**false**, **true**}
- **false** = 0
- **true** = 1 dar și orice întreg nenul
- Operații: **!** **&&** **||** **==** **!=**
- Declarații posibile:

```
bool x = 7;    // x devine "true"  
int y = true;  // y devine 1
```

# Expresii logice

***expresie\_relationala ::=***

*expr < expr / expr > expr*

*/ expr <= expr / expr >= expr*

*/ expr == expr / expr != expr*

***expresie\_logica ::=***

*! expr*

*/ expr || expr*

*/ expr && expr*

# Valoarea expresiilor relaționale

$a-b$	$a < b$	$a > b$	$a \leq b$	$a \geq b$	$a == b$	$a != b$
<b>pozitiv</b>	0	1	0	1	0	1
<b>zero</b>	0	0	1	1	1	0
<b>negativ</b>	1	0	1	0	0	1

# Valoarea expresiilor logice ||

<b>exp1</b>	<b>exp2</b>	<b>exp1    exp2</b>
<b>&lt;&gt; 0</b>	nu se evaluează	<b>1</b>
<b>= 0</b>	se evaluează	1, dacă exp2 <> 0 0, dacă exp2 = 0

# Valoarea expresiilor logice &&

<b>exp1</b>	<b>exp2</b>	<b>exp1 &amp;&amp; exp2</b>
= 0	nu se evaluează	0
<> 0	se evaluează	1, dacă exp2 <> 0 0, dacă exp2 = 0

# Exemple

- O condiție de forma  $a \leq x \leq b$  se scrie în limbajul C++:

```
(x >= a) && (x <= b)
```

```
(a <= x) && (x <= b)
```

- O condiție de forma  $a > x$  sau  $x > b$  se scrie în limbajul C++:

```
(x < a) || (x > b)
```

```
!(x >= a && x <= b)
```

# Tipul **void**

- Conversia în tip **void** a unei expresii semnifică faptul că valoarea sa este ignorată
- Utilizat pentru tipul `pointer`; nu se face controlul tipului la un `pointer` de tip **void**
- Utilizat pentru funcții fără valoare returnată sau pentru funcții fără parametri
- Este un tip incomplet ce nu poate fi completat

# Tipul **void**

- Conversia în tip **void** a unei expresii semnifică faptul că valoarea sa este ignorată
- Utilizat pentru tipul `pointer`; nu se face controlul tipului la un `pointer` de tip **void**
- Utilizat pentru funcții fără valoare returnată sau pentru funcții fără parametri
- Este un tip incomplet ce nu poate fi completat