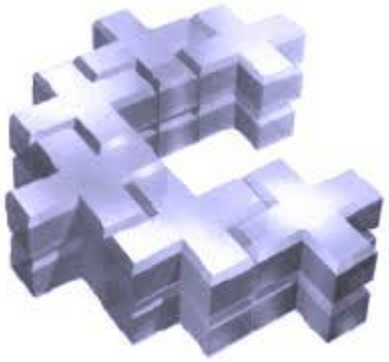
Elementele de bază ale limbajului de programare C++



Sumar

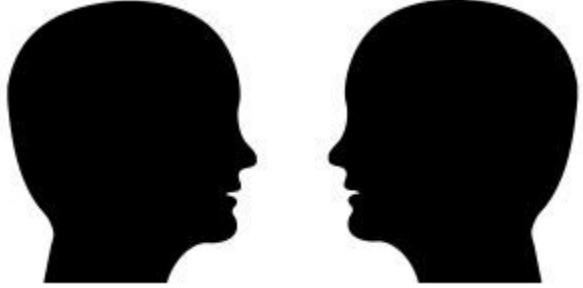
1. Competenţe

Competenţe generale

- *implementarea algoritmilor într-un limbaj de programare*
- *aplicarea algoritmilor fundamentali în prelucrarea datelor*

Competenţe specifice

- *transcrierea algoritmilor din limbaj pseudocod în limbaj de programare*
- *elaborarea unui algoritm de rezolvare a unor probleme din aria curriculară a specialităţii*
- *alegerea unui algoritm eficient de rezolvare a unei probleme*



2. Noţiuni introductive

Noţiuni introductive

Orice limbaj constituie un mijloc de comunicare între două entităţi: emiţătorul şi receptorul.

În general limbajele sunt de două tipuri:

- limbaje naturale;
- limbaje artificiale.

Limbajele naturale s-au constituit de-a lungul timpului, în procesul conlucrării membrilor societăţii.

Limbajele artificiale au fost şi sunt create pentru comunicarea într-un domeniu particular de activitate.

Noţiuni introductive

- Limbajele de programare fac parte din categoria limbajelor artificiale, fiind utilizate în procesul de comunicare om-calculator.

- ***Un limbaj de programare reprezintă un mijloc de comunicare între programator şi calculator.***

Noţiuni introductive

Repere istorice în evoluţia limbajelor de programare:

- 1955 – FORTRAN (FORmula TRANslation)
- 1960 – ALGOL (ALGOrithmic Language)
- 1960 – COBOL (COmmon Business Oriented Language)
- 1971 – Pascal (Blaise PASCAL)
- 1972 – C
- 1980 – C++
- 1995 – Java

Noţiuni introductive

Limbajul de programare C++

La începutul anilor 70 a apărut limbajul C – creaţia lui Dennis Ritchie şi Brain Kernighan.

Limbajul C++ este creaţia lui Bjarne Stroustrup şi reprezintă o extensie a limbajului C care permite programarea pe obiecte.

Noţiuni introductive

Realizarea unui program scris în C++ necesită parcurgerea a patru etape:

- ***editare*** – scrierea programului sursă, prin crearea unui fişier cu extensia cpp;
- ***compilare*** – se aduce în memoria internă programul sursă, se verifică erori şi se converteşte acest program în program obiect, având extensia obj;
- ***link-editare*** – se leagă programul obiect cu bibliotecile de sistem şi se transformă într-un program executabil având extensia exe;
- ***execuţie*** – se lansează în execuţie programul obiect: se efectuează citirea datelor, calculele şi scrierea rezultatelor, formându-se fişierul.

3. Structura generală a unui program C++

Structura generală a unui program C++

- un program C++ este constituit dintr-o succesiune de module, denumite ***funcţii***
- una dintre aceste funcţii este funcţia principală, denumită ***main()***
- ***main()*** este o funcţie specială, care trebuie să apară obligatoriu o singură dată în orice program C++
- execuţia oricărui program începe cu funcţia ***main()***
- o funcţie este constituită din ***antet şi corp***
- ***antetul*** funcţiei conţine numele funcţiei, tipul rezultatului pe care îl calculează funcţia şi o listă de parametri prin care funcţia comunică cu exteriorul ei, încadrată între paranteze rotunde
- ***corpul*** funcţiei conține declarații și instrucțiuni care specifică prelucrările realizate de funcția respectivă

Structura generală a unui program C++

Forma funcţiei main

```
int main()
 {
        . . . .
        return 0;
}
```

Instrucţiunea *return* este utilizată pentru a încheia execuţia unei funcţii şi a returna valoarea expresiei specificate în instrucţiunea return ca valoare a funcţiei.

4. Elementele de limbaj

Limbajul C++ este caracterizat de:

- ***sintaxă*** – este formată din totalitatea regulilor de scriere corectă a programelor;
- ***semantică*** – reprezintă semnificaţia construcţiilor corecte din punct de vedere sintactic;
- ***vocabular*** – este format din totalitatea cuvintelor care pot fi folosite într-un program.

5. Vocabularul limbajului C++

Vocabularul limbajului C++ este format din:

- setul de caractere;
- identificatori;
- cuvinte cheie;
- comentarii;
- separatori.

Vocabularul limbajului C++

a. ***Setul de caractere***

Setul de caractere utilizat pentru scrierea programelor C++ este setul de caractere al codului ASCII.

Codul ASCII este format din:

- literele mari şi mici ale alfabetului latin (A-Z, a-z);
- cifrele sistemului de numeraţie zecimal (0-9);
- caracterele speciale (blank, +, *, %, =, {, !, #, etc.).

Vocabularul limbajului C++

b. Identificatori

Identificatorii (numele) au rolul de a denumi elemente ale programului precum constante, variabile, funcţii etc.

Identificatorii:

- reprezintă o secvenţă de litere, cifre şi _ (linia de subliniere) care trebuie să înceapă cu _ sau cu o literă;
- nu pot fi cuvinte cheie (rezervate) ale limbajului.

Exemple -corecte	**Contraexemple -incorecte**
suma	suma 1
Suma	1suma
suma1	suma+1
suma_1	suma&nr
_suma	suma nr

Vocabularul limbajului C++

c. Cuvinte cheie (rezervate)

Cuvintele cheie (keywords) sunt cuvinte care au un înţeles bine definit şi nu pot fi folosite în alt context.

Exemple

void	**default**	**for**	**struct**
break	**do**	**if**	**switch**
case	**double**	**int**	**unsigned**
char	**else**	**long**	**while**
const	**float**	**return**	

Vocabularul limbajului C++

d. Comentarii

Pentru ca un program să fie uşor de înţeles se folosesc comentariile. Acestea sunt texte care vor fi ignorate de compilator, dar au rolul de a explicita pentru programator anumite secvenţe de program.

// comentariu

sau

/*comentariu

comentariu

.....*/

Vocabularul limbajului C++

e. Separatori

Separatorii se folosesc pentru a delimita unităţile sintactice. Separatori:

- blank
- TAB
- caracterele de control CR (*Carriage Return*=pozitioneaza cursorul in coloana 1 a noului rand*)*+LF(*Line Feed* =rand nou) generate de tasta Enter
- virgula

6. Tipuri simple de date

Tipuri simple de date (standard)

Prin ***date*** se înţelege, în general, tot ceea ce este prelucrat de un calculator. Fiecare dată are un anumit tip.

Un ***tip de date*** defineşte:

- mulţimea valorilor pe care le pot lua datele de tipul respectiv;
- modul de reprezentare a acestora în memorie;
- operaţiile care se pot efectua cu datele respective.

Clasificarea tipurilor de date:

- tipuri de date predefinite - asociate cu un cuvânt cheie, utilizat în declaraţie;
- tipuri de date definite de utilizator.

Tipuri simple de date

Tipuri standard în C++:

- **int şi long – pentru memorarea numerelor întregi;**
- **float şi double pentru memorarea numerelor reale;**
- **char – pentru memorarea caracterelor;**
- **void –** pentru tip neprecizat.

Tipul ***void*** este un tip special, pentru care mulţimea valorilor este vidă. Acest tip se utilizează atunci când este necesar să specificăm absenţa oricărei valori. De exemplu, poate fi utilizat pentru a specifica tipul unei funcţii care nu returnează niciun rezultat.

Tipuri simple de date

- Tipuri standard în C++. Domeniul de valori şi dimensiunea memoriei ocupate:

Tipul	NUME TIP	DIMENSIUNE IN BITI	DOMENIU
Tipul întreg	unsigned int	16	0..65535
	short int	16	-32768..32767
	int	16	-32768..32767
	unsigned long	32	0..4294967295
	long	32	-2147483648..2147483647
Tipul real	float	32	3.4*pow(10,38)
	double	64	1.7*pow(10.308)
	long double	80	1.1*pow(10,4932)
Tipul caracter	unsigned char	8	0..255
	char	8	-128..127

7. Constante şi variabile

Constante şi variabile

O categorie aparte de date o reprezintă constantele şi variabilele.

Constantele

• constanta are un tip şi o valoare fixă pe toată durata execuţiei programului care o conţine;

• tipul şi valoarea unei constante se definesc prin caracterele care compun constanta respectivă.

Constantele se clasifică astfel:

• numerice: - întregi - reale

• caracter

• şir de caractere

Constante şi variabile

Declararea constantelor

Sintaxa:

const [tip_dată] nume=valoare;

unde:

- **const este un cuvânt cheie care înseamnă definirea unei constante simbolice;**
- **tip_dată precizează tipul constante (poate lipsi);**
- **nume este identificatorul constantei;**
- **valoare este valoarea constantei.**

Exemple

```
const int a=0;
const int x=-5;
const b=0;
const float PI=3.14;
const char a='a';
const char sir[]="info";
```

Constante şi variabile

Variabile

- nume asociat cu una sau mai multe locaţii de memorie;
- valoarea păstrată în aceste locaţii se poate modifica în cursul execuţiei programului;
- trebuie declarate – se specifică tipul şi numele.

Constante şi variabile

Declararea variabilelor

Sintaxa:

tip_dată nume;

unde:

•tip_dată precizează tipul datei memorate în variabila de memorie;

• nume este identificatorul variabilei de memorie.

 Exemple

```
int a;
int x,y;
char b;
int a,b=1, c=2;
float d=1;
float e=1.234;
char f='a';
long x1,x2;
unsigned int p,q;
char sir[]="info";
```

8. Operatori şi expresii

Operatori şi expresii ***Operatori Operatorii sunt caractere speciale care indică operaţia care se efectuează în cadrul unui program. Clasificarea operatorilor:***

- operatori aritmetici;
- operatori relaţionali;
- operatori de egalitate;
- operatori de incrementare şi decrementare;
- operatori logici;
- operatori de atribuire;
- operatorul „**,**“ **(virgulă);**
- operatorul de conversie explicită.

Operatori şi expresii

a. Operatori aritmetici

- **- minus (unar) – pentru semn**
- **+ plus (unar) – pentru semn**
- **+ (binar) – adunare**
- **- (binar) – scădere**
- *** (binar) – înmulţire**
- **/ (binar) – împărţire întreagă**
- **% (binar) – restul împărţirii întregi**

 Exemple

```
int a=3,b=4,p,c,r;
p=a*b;
c=a/b+p;
r=a%b;
```

Operatori şi expresii

b. Operatori de comparaţie (relaţionali)

< mai mic

> mai mare

<= mai mic sau egal

>= mai mare sau egal

Rezultatul obţinut în cazul aplicării unuia dintre operatorii relaţionali este *true sau false.*

Exemple

```
2<=5
4<3
int x=4,y=5,c;
c=x>y;
```

Operatori şi expresii

c. Operatori de egalitate

== egal

!= diferit

Rezultatul obţinut în cazul aplicării unuia dintre operatorii de egalitate este *true sau false.*

- ***Exemple***

```
3==3
5==8
3!=6
4!=4
int a=8,b=8,x;
x=a==b;
```

Operatori şi expresii

d. Operatori de incrementare şi decrementare

++ incrementare (adună 1)

-- decrementare (scade 1)

Exemple

```
int a=8,b=4,c=6,x;
a++; //a=9
x=b--; //x=4, b=3
x=++c; //x=7, c=7
```

Operatori şi expresii

e. Operatori logici

&& ŞI logic

|| SAU logic

! negaţie

p	q	p && q
0	0	0
0	1	0
1	0	0
1	1	1

p	q	p || q
0	0	0
0	1	1
1	0	1
1	1	1

p	!p
0	1
1	0

Rezultatul obţinut în cazul aplicării unuia dintre operatorii logini este true sau false.

Exemple

a<=b && a<=c

a>5 || b<8

!(a==b)

Operatori şi expresii

f. Operatori de atribuire

= egal

***=**

/=

%=

+=

-=

Exemple

```
int a=2,b=3,c=4;
a=b;
b+=a; //b=b+a
 c=b=a;
```

Operatori şi expresii

g. Operatorul ‘,’ (virgulă)

Separă mai multe expresii.

Exemple

```
int a=1, b=5;
float c;
c=a=b+1,a=c+2,b=b+1;
//b+1=6; a=6; c=6
//a=6+2=8;
//b=5+1=6;
```

Operatori şi expresii

h. Operatorul de conversie explicită Pentru ca un operand să intre în calcul convertit aşa cum ne dorim (nu implicit) înaintea operandului se trece tipul său. Exemple

```
float x=25.79; //x=25.79
int y;
y=x;                    //y=25
x=(int)x;               //x=25
x=int(x);               //x=25
float a=8, b=3, c;
c=a/b;                  //c=2.66667
```

Operatori şi expresii

- ***Prioritatea operatorilor***

Clasă 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
relaţionali	< <= > >=	de la stânga la dreapta
de egalitate	== !=	de la stânga la dreapta
logici	&&	de la stânga la dreapta
logici	||	de la stânga la dreapta
atribuire şi aritmetici binari	= *= /= %= += -=	de la dreapta la stânga

Operatori şi expresii

Expresii

O expresie este alcătuită din unul sau mai mulţi operanzi legaţi între ei prin operatori. Operanzii pot fi constante, variabile sau funcţii.

Operanzii reprezintă valorile care intră în calcul, iar operatorii desemnează operaţiile care se execută în cadrul expresiei.

expresie = operatori + operanzi

Tipul unei expresii reprezintă tipul valorii expresiei.

Expresiile se împart în două categorii:

- expresii aritmetice;
- expresii logice.

Operatori şi expresii

a. Expresii aritmetice

- expresiile aritmetice sunt cele care efectuează operaţii aritmetice având ca rezultat un număr
  Exemple

```
int x=7, y=2, r;
r=x/y;                          //r=3
float x=7, y=2, r;
r=x/y;                          //r=3.5
int a; a=25/2*4-3+7/2;          //a=48
```

Operatori şi expresii

b. Expresii logice

- o expresie logică descrie o condiţie
- valoarea unei expresii logice reprezintă valoarea de adevăr a expresiei aferente
- o condiţie poate fi falsă/false (valoarea 0) sau adevărată/true (o valoare diferită de 0)

Exemple int x=7, y=2;

```
x>=y        //true
x!=y        //true
x<y         //false
```

9. Operaţii de citire şi scriere

Operaţii de citire şi scriere

În limbajul C++ operaţiile de introducere şi extragere date se execută prin fluxurile de date.

Un *flux de date (stream)* reprezintă fluxul datelor de la sursă (de exemplu tastatură) la destinaţie (de exemplu ecranul monitorului).

Prin fluxurile de date echipamentele periferice de intrare-ieşire sunt conectate la programul C++.

Fluxuri de date standard

1. flux de date de intrare (cin);

2. flux de date de ieşire (cout).

Pentru operaţiile de citire şi scriere se folosesc instrucţiunile expresie prin care se creează fluxurile de date, cu ajutorul operatorilor >> şi <<.

Operaţii de citire şi scriere

a. Flux de date de intrare (cin)

- conectează tastatura la program
- execută operaţii de citire
- datele de intrare sunt furnizate programului
- datele sunt păstrate în variabile de memorie
- ***cin reprezintă tastatura***
- operatorul de intrare **>>** înseamnă transmiterea unei valori de la tastatură

Sintaxa:

cin>>nume_var;

sau

cin>>nume_var1>>nume_var2 >> … >>nume_varn;

Operaţii de citire şi scriere

Exemplu

```
int x=7,y=2,z=4;
cin>>x;
cin>>y;
cin>>z;
// considerăm că se introduc de la tastatură
   valorile 10, 20 şi 30
```

Operaţii de citire şi scriere

2. Flux de date de ieşire (cout)

- conectează monitorul la program
- execută operaţii de scriere
- datele de ieşire sunt furnizate de program
- datele sunt transmise către monitor
- ***cout*** reprezintă monitorul
- operatorul de ieşire **<<** înseamnă transmiterea unei valori către monitor

Sintaxa:

cout<<nume_var|constantă;

sau

cout<<nume_var1|constantă1<< nume_var 2|constantă2<< … <<nume_varn|constantăn;

Operaţii de citire şi scriere

Exemplu

```
int x=7,y=2,z=4;
 cout<<x;
cout<<y;
cout<<z;
```

se va afişa: 724

iar pentru

```
cout<<x<<“ “;
cout<<10<<endl;
cout<<z;
```

se va afişa: 7 10
4