

5. Aplicații cu funcții și intrări/ieșiri în C/C++. (C/C++ applications using functions and input/output operations)

1. Obiective:

- Înțelegerea noțiunii de *flux (stream)*
- Înțelegerea sintaxei funcțiilor *printf()* și *scanf()*
- Utilizarea specificatorilor de format pentru *printf()* și *scanf()*
- Înțelegerea sintaxei operatorilor *>>* (*extracție*) și *<<* (*inserție*)
- Scrierea și testarea unor programe simple ce folosesc intrări/ieșiri C/C++

1. Objectives:

- Understanding the *stream* concept
- Understanding the *printf()* and *scanf()* syntax
- Understanding the usage of *printf()* and *scanf()* format specifiers
- Understanding the extraction *>>* and insertion *<<* operators' syntax
- Writing and testing some simple programs that use C/C++ I/O

2. Breviar teoretic

2.1. Flux (stream)

Sistemul de intrări-ieșiri din C/C++ operează prin *stream*-uri (fluxuri). Un *stream (flux)* este un dispozitiv logic, care fie produce, fie consumă informație. Altfel spus, reprezintă totalitatea modalităților de realizare a unor operații de citire sau scriere.

Transferurile cu dispozitivele periferice (consolă, disc, imprimantă) se fac prin intermediul fluxurilor și prin intermediul sistemului de operare. În acest mod, detaliile funcționale ale dispozitivelor periferice pot fi ignorate la nivelul programului.

Un flux poate fi de intrare, de ieșire sau bidirecțional.

Există fluxuri predefinite (standard) care se creează automat la lansarea în execuție a unui program și sunt închise automat la încheierea execuției programului.

2.2. Intrări/ieșiri C

Cele mai importante fluxuri predefinite (standard) din limbajul C sunt următoarele:

- *stdin*: intrare standard, asociată consolei în intrare, adică tastaturii;
- *stdout*: ieșire standard, asociată consolei în ieșire, adică ecranului;

În limbajul C, sistemul de intrări/ieșiri nu este o parte a limbajului, ci este adăugat printr-un set de funcții din biblioteca standard. Funcțiile de intrare/ieșire pentru consolă utilizează implicit dispozitivele predefinite *stdin (tastatura)* și *stdout (ecranul)*, în mod transparent pentru programator.

Funcția *printf()* :

- permite formatarea și afișarea de caractere și valori către ieșirea standard, *stdout*.
- se apelează astfel: ***printf(format[,arg_i])***;
 - unde *format* este un șir de caractere care definește textele, secvențele escape și formatele de scriere a datelor precizate prin specificatori de format care se scriu în caz că există *arg_i*;
 - *arg_i*, sunt argumente care trebuie să corespundă specificatorilor de format corespunzând specificatorului *i*: de forma
%[-,+] *dimensiune_minimă_câmp.precizie_afișare* *format_tip_dată*

Datele gestionate de către *printf()* sunt supuse unor transformări din cauza existenței unui format intern (ce depinde de tipul datelor) și a altuia extern a datelor (uzual ASCII). Specificatorii de format definesc aceste conversii.

Pentru afișarea datelor de tip `wchar_t` se utilizează o funcția **wprintf()** - versiune a lui `printf()` definită în biblioteca `<wchar>`

Specificatorii de format încep totdeauna cu caracterul `%`. Formatele specifice utilizate în *printf()* sunt:

- `%c` afișare caracter unic; valoarea lui este interpretată ca fiind codul ASCII al caracterului;
- `%s` afișare șir de caractere până la caracterul NULL;
- `%d` afișare număr întreg în baza zece cu semn;
- `%i` afișare număr întreg în baza zece cu semn;
- `%u` afișare număr întreg în baza zece fără semn realizând conversia unei date binare de tip *unsigned* în zecimal;
- `%f(F)` afișare număr real, notația zecimală realizând conversia datei de tip *float* sau *double* la forma, *întreg.fractionar*;
- `%e(E)` afișare număr real, notația exponențială realizând conversia datei de tip *float* sau *double* la forma, *întreg.fractionarE[+,-]exp*;
- `%g(G)` afișare număr real, cea mai scurtă reprezentare dintre `%f` și `%e`;
- `%a(A)` afișare număr real în hexazecimal
- `%x(X)` afișare număr hexazecimal întreg fără semn convertind datele *int* sau *unsigned*;
- `%o` afișare număr octal întreg fără semn convertind datele *int* sau *unsigned*;
- `%p` afișarea unui pointer la *void* (ori ce tip de dată) sub forma unei adrese compuse din segment și offset în afișare cu cifre hexa;
- `%n` înscrie în variabila de tip întreg a cărei adresă e dată ca argument, numărul de caractere scrise deja în flux cu `printf()` până la întâlnirea lui `%n` (implicit dezactivat în Visual Studio)

- **lc** **wchar_t** wide character pentru *wprintf*
- **ls** **wchar_t *** wide string pentru *wprintf*

În plus, mai pot fi utilizate următoarele *prefixe*:

- l* - cu *d, i, o, u, x, X* pentru date de tip *long*;
- l, L* - cu *e, f, g, E, G, a, A* pentru date de tip *double*;
- h* - cu *d, i, o, u, x, X*, pentru date de tip *short*.

Pentru a specifica:

- dimensiunea câmpului de afișare și precizia de afișare a datelor, în cadrul datelor, specificatorul de format va arăta astfel:

`%[-,+]a.bf` unde:

- `%` este semnul începutului specificatorului de format;
- indică că alinierea are loc la stânga;
- + Valorile pozitive sunt precedate de +
- a* precizează dimensiunea minimă a câmpului (inclusiv punctul care se scrie dacă este cazul);
- b* dă precizia de afișare;

f arată formatul tipului de dată specific, *f* pentru real, *d* pentru întreg, *s* pentru șir.

Semnificatia preciziei este prezentata în tabelul urmator :

Tip	Format (.b)
d, i, o, x, X	se afișează minimum <i>b</i> digiți, eventual completați la stânga cu 0 (implicit: <i>b</i> =1)
e, E, f, F, a, A	se afișează maximum <i>b</i> zecimale (implicit <i>b</i> =6)
g, G	se afișează maximum <i>b</i> cifre semnificative (implicit: toate cifrele semnificative)
s	se afișează maximum <i>b</i> caractere din șir (implicit: toate caracterele)

Observații:

1. precizia poate fi stabilită și folosind caracterul '*' după caracterul '.', valoarea preciziei fiind dată printr-un argument suplimentar al funcției *printf()*, care precede argumentul ce trebuie formatat.
2. Dimensiunea minimă poate fi specificată în mod similar, folosind caracterul '*' între % și specificatorul de format, valoarea dimensiunii minime a câmpului fiind specificată printr-un argument suplimentar plasat înaintea argumentului care va fi formatat.

Funcția *scanf()*:

- permite introducerea de date tastate la terminalul standard de intrare *stdin*, date specificate de argumentele *arg_i* sub controlul unor formate.
- se apelează astfel: ***scanf(format [,arg_i]);***
 - unde *format* este un șir de caractere care dă formatele datelor și eventual textele aflate la *stdin*; caracterele albe sunt neglijate; în rest sunt *specificatori de format* și alte caractere care trebuie să existe la intrare în pozițiile corespunzătoare, caractere în general folosite la efectuarea de controale asupra datelor citite.
 - *arg_i*, sunt argumente care corespund adreselor zonelor în care se păstrează datele citite după ce au fost convertite din formatul lor extern în cel intern corespunzător, de forma: *%format_tip_dată*

Specificatorii de format pentru *scanf()*:

Tip	Intrare
d	întreg zecimal
D	întreg zecimal
o	întreg octal
O	întreg octal
x	întreg hexazecimal
X	întreg hexazecimal
i	întreg (d, o, x)
I	întreg (D, O, X)
u	întreg zecimal fără semn
U	întreg zecimal fără semn
e, E, f, F, g, G	număr real
c	caracter
s	șir de caractere
lc	wide character pentru <i>wscanf</i>
ls	wide string pentru <i>wscanf</i>

Între caracterul % și literele specifice specificatorului de format se mai pot utiliza:

- *un caracter ** opțional, precizând faptul că ceea ce urmează a fi preluat și interpretat este funcție de tipul specificat, dar nu este memorat;
- *un șir de cifre* opțional care definește lungimea maximă a câmpului din care se citește data sub controlul formatului respectiv.

Funcția *scanf()* citește toate câmpurile care corespund specificatorului de format, inclusiv eventuale texte prezente în câmpul format. În caz de eroare, citirea se oprește, funcția *scanf()* returnând numărul de câmpuri citite corect.

Noi elemente legate de operațiile de intrare/ieșire cu *printf()/scanf()* au fost introduse în *cstdio* și *cinttypes* (*inttypes.h*) în C++0x/1y/2z (*long long int* (*long long int -%lld – pentru signed, unsigned long long int -%llu – pentru unsigned, etc.*)), vedeți la adresele :

<http://www.cplusplus.com/reference/cstdio/printf/>

<http://www.cplusplus.com/reference/cstdio/scanf/>

În C++0x/1y/2z pot fi utilizate expresii regulate pentru a permite citirea caracterelor, chiar dacă se întâlnesc caractere de spațiere.

```
char name[20]="";
scanf ("%[^\\n]%"c", name);
```

[^\\n] – specifică citirea caracterelor până la sfârșit de linie.

%*c – caracterul * indică faptul că se extrage din flux caracterul newline (deci nu va afecta citiri ulterioare), dar acesta nu va fi memorat.

```
scanf ("%[^\\n]s",name);
\\n- setează delimitatorul pentru șirul citit
```

2.3. VC++ *scanf_s()* și alte funcții asociate

- *scanf_s()* citește date formate de la intrarea standard. Există mai multe versiuni ale lui *scanf_s()*, *_scanf_l()*, *wscanf()*, *_wscanf_l()* (ultimele două pentru citire date de tip *wchar_t*) care au îmbunătățiri de securitate, cum e descris în “Security Features in the CRT”.
- *int scanf_s(const char *format [,argument]...);*
- *int _scanf_s_l(const char *format, locale_t locale [, argument]...); etc.*

Parametrii:

- *format*, șir de control format.
- *argument*, argumente.
- *locale*, variabile locale de folosit

Exemplu:

```
char s[10];
scanf_s ("%9s", s, (unsigned)_countof(s)); // buferul are dimensiunea 10, numărul maxim de
// caractere citite este 9
```

```
#define _CRT_SECURE_NO_WARNINGS
```

Directiva e folosită pentru a considera funcția standard C/C++ *scanf()* fără a avea mesaje de incompatibilitate.

2.4 Intrări/ieșiri C++

Limbajul C++ conține toate rutinele din biblioteca de intrări/ieșiri a limbajului C, dar ne pune la dispoziție și un sistem propriu de intrări/ieșiri orientat pe obiecte, implementat prin așa numita bibliotecă *iostream*.

În limbajul C++ sunt predefinite dispozitivele logice de *intrare/ieșire* standard similare celor din C:

- *cin* (console input): dispozitiv de intrare consolă, asociat tastaturii (echivalentul lui *stdin*);
- *cout* (console output): dispozitiv de ieșire consolă, asociat monitorului (echivalentul lui *stdout*).

Transferul informației cu formatare se poate face cu operatori (fac parte din limbaj) specializați:

- `>>` (extracție) pentru intrare (de la *cin*): *cin >> var;*
- `<<` (inserție) pentru ieșire (către *cout*): *cout << var;*

Sunt posibile operații multiple:

- *cout << var₁ << var₂...<< var_n;*
- *cin >> var₁ >> var₂...>> var_n;*

chiar dacă variabilele implicate au tipuri diferite.

Utilizarea dispozitivelor și operatorilor de intrare/ieșire C++ impune includerea fișierului antet *iostream* și a spațiului de nume *std*. (în alte medii *iostream.h*)

Acești operatori nu necesită specificatori de format pentru fiecare tip de dată deoarece se folosește un format implicit. Particularizarea formatului este posibilă, dar este mai dificilă față de lucrul cu funcțiile *printf()/scanf()*.

3. Exemple:

Exemplul 1: program ce afișează un întreg în zecimal, octal și hexazecimal.

```
// directive preprocesor de includere
```

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

```
// defînire constanta simbolică
```

```
#define V 12345
```

```
//constexpr auto V1 = 12345;
```

```
int main(void)
```

```
{
```

```
    printf("Zecimal: %d\n", V);
```

```
    printf("Octal: %o\n", V);
```

```
    printf("Hexazecimal: %x\n", V);
```

```
    printf("Afișare întreg într-un câmp cu dimensiunea 10:%*d\n", 10, V);
```

```
    puts("Acționați o tastă pentru a continua");
```

```
    return 0;
```

```
}//end main
```

Exemplul 2: program ce citește un caracter folosind funcția *scanf()*; afișează codul ASCII al caracterului respectiv.

```
#define _CRT_SECURE_NO_WARNINGS
// directive preprocesor de includere
#include <stdio.h>

int main(void){
// declaratii locale
char car;
    printf("\nIntroduceti un caracter : ");
    scanf("%c", &car); //un spatiu inainte de %, in special daca se fac alte citiri in prealabil
    printf("\n Codul ASCII al caracterului introdus : %d\n", car);
    return 0;
} //end main
```

Exemplul 3: program ce afișează valoarea Pi în diferite formate.

```
// directive preprocesor de includere
#ifndef _USE_MATH_DEFINES
#define _USE_MATH_DEFINES
#endif

#include <stdio.h>
#include <math.h>

int main(void){
    printf("\nPi=%f",M_PI);           // Pi=3.141593
    printf("\nPi=%-10f",M_PI);       // Pi=3.141593, pe 10 pozitii, aliniere la stanga
    printf("\nPi=%10f",M_PI);       // Pi=3.141593, pe 10 pozitii, aliniere la dreapta
    printf("\nPi=%e",M_PI);          // Pi=3.141593e+00 (notatia stiintifica)
    printf("\nPi=%g",M_PI);          // Pi=3.14159, afisarea implicita este pe 6 digiti
    printf("\nPi=%06.2f",M_PI);      // Pi=003.14, pe 6 pozitii, din care 2 zecimale si
                                    // completare cu zerouri la stanga
    printf("\nPi=%10.*f",2,M_PI);    // Pi=3.14, pe 10 pozitii si precizie 2, aliniere la dreapta
    printf("\nPi=%a",M_PI);          // Pi=0x1.921fb5p+1, afisare in hexazecimal cu litere mici
    printf("\nPi=%A\n",M_PI);        // Pi=0X1.9220P+1, afisare in hexazecimal cu majuscule, precizie 4
}
```

Exemplul 4: utilizarea specificatorului %n si afisarea adresei unei variabile folosind %p

```
#include<stdio.h>
int main()
{
    int i;
    _set_printf_count_output( 1 ); // activare %n prin apelul functiei cu un parametru diferit de 0
    printf("12345%n6789\n", &i ); // i primeste numarul caracterelor scrise deja pe ecran cu
                                    // printf() pana la %n, adica i=5. Obs: Se specifica adresa lui i
    printf("i = %d\n", i ); // afisarea valorii lui i
    printf("Adresa lui i=%#p\n\n", &i); // %p pentru afisarea adresei si # pentru afisare 0X //inaintea
                                    // adresei reprezentata in hexazecimal
    getchar(); // asteapta un caracter de la tastatura – functie din biblioteca stdio.h
    return 0;
}
```

Exemplul 5: program ce citește un flotant și un șir de caractere folosind *scanf_s()*:

```
//#define _CRT_SECURE_NO_WARNINGS
#include <stdio>
#include <stdlib> // _countof

#define MAX 10

int main() {
    char buff[MAX];
    float a;
    printf("\nEnter a float and string:");
    scanf_s("%f", &a); // Atentie ! Nu %F
    scanf_s("%s", buff, _countof(buff)); //sizeof(buff)
    // scanf("%s", buff);
    printf("\nFloat is: %6.2f\n", a); //Nu %6.2F
    printf(buff);
    return 0;
}
```

Exemplul 6: program ce citește numele, prenumele și anul nașterii unei persoane separate prin spații albe, după care afișează informațiile obținute pe un singur rând (intrări - ieșiri C).

```
#define _CRT_SECURE_NO_WARNINGS
// directive preprocesor de includere
#include <stdio.h>

int main(void)
{
    // declaratii locale
    char nume[50], prenume[50];
    int an;

    printf("\nIntroduceti numele: ");
    scanf("%s", nume);
    printf("\nIntroduceti prenumele: ");
    scanf("%s", prenume);
    printf("\nIntroduceti anul nasterii: ");
    scanf("%d", &an);

    printf("\nDatele persoanei: %s, %s, %d\n", nume, prenume, an);
    return 0;
} //end main
```

Exemplul 7: program care utilizează expresii regulate pentru citirea șirurilor de caractere cu spații

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio>
#define DIM 20
```

```

int main() {
    char name[DIM] = "";
    printf("\nEnter a string with space: ");
    scanf("%[^\\n]%*c", name);
    printf(name);
    printf("\nEnter a new string with space: ");
    scanf("%[^\\n]s", name);
    printf(name);
    return 0;
}

```

Exemplul 8: citire/afisare date de tipul `wchar_t`

```

#include<cstdio>
#include <wchar> // pentru functii si tipuri wide
#include<stdlib.h>

```

```

int main() {
    wchar_t ch;
    wchar_t name[] = L"Students";
    wchar_t message[30];
    printf("\nEnter a wide character: ");
    wscanf_s(L"%lc", &ch, sizeof(ch));
    wprintf(L"character = %c \n", ch);

    wprintf(L"HELLO %ls \n", name);
    printf("\nEnter a message: ");
    wscanf_s(L"%ls", message, (unsigned)_countof(message));
    wprintf(L"Message: %ls \n", message);
}

```

Exemplul 9: program ce citește numele, prenumele și anul nașterii unei persoane separate prin spații albe; afișează informațiile obținute pe un singur rând (intrări/ieșiri C++).

a)

// directive preprocesor de includere

```

#include <iostream>
using namespace std;

```

```

int main(void)
{
    // declaratii locale
    char nume[255], prenume[255];
    int an;

```

```

    cout << "\nIntroduceti numele: ";
    cin >> nume;
    cout << "\nIntroduceti prenumele: ";
    cin >> prenume;

```



```

    cout << "\nIntroduceti anul nasterii: ";
    cin >> an;
    cout << endl << "Datele persoanei: " << nume << " " << prenume << ", " << an << "\n";
    return 0;
} //end main

```

b)

//Siruri de caractere wide

#include <iostream>

using namespace std;

const int MAX = 255;

int main() {

wchar_t wnume[MAX], wprenume[MAX];

int an;

wcout << L"\nIntroduceti numele- wide string: ";

wcin >> wnume;

wcout << L"\nIntroduceti prenumele- wide string: ";

wcin >> wprenume;

wcout << L"\nIntroduceti anul nasterii: ";

cin >> an;

wcout << endl << L"Datele persoanei: " << wnume << " " << wprenume << ", "
<< an << endl;

} //end main

Exemplul 10: program ce citește o dată calendaristică sub forma: zz ll aa și o afișează sub forma: aa/ll/zz.

// directive preprocesor de includere

#include <iostream>

using namespace std;

int main(void)

{

// declaratii locale

int zi, luna, an;

cout << "\nIntroduceti data (zz ll aa): ";

cin >> zi >> luna >> an;

cout << endl << an << "/" << luna << "/" << zi << "\n";

return 0;

} //end main

Exemplul 11: program ce utilizeaza variabile de tip *auto*

//auto type

```

#include <iostream>
using namespace std;

int main() {
    auto x = 4; // x -> int
    auto y = 3.37; // y -> double
    auto z = 'x'; // z -> char
    cout << "\n The variables are initial: \t x= " << x << " y= " << y << " z= " << z << endl;
    cout << "\n Enter 3 variables (int, double, char): \n";
    cin >> x >> y >> z;
    cout << "\n The new variables are: \t x= " << x << " y= " << y << " z= " << z << endl;
}

```

4. Întrebări:

1. Ce este un flux (stream) ?
2. Care sunt cele mai importante fluxuri în limbajul C ?
3. Care sunt cele mai importante fluxuri în limbajul C++ ?
4. Care sunt principalele funcții ce permit efectuarea operațiilor de intrare/ieșire în limbajul C ?
5. Care sunt operatorii ce permit efectuarea operațiilor de intrare/ieșire în limbajul C++ ?

5. Teme:

1. Realizați o aplicație care citește de la intrarea standard două valori pentru variabilele R1 și R2, reprezentând valoarea rezistențelor R1 și R2, iar apoi apelează funcții ce calculează rezistența echivalentă grupării serie, respectiv grupării paralel, după care afișează valorile returnate cu 3 zecimale și aliniere la dreapta.
2. Scrieți o aplicație care citește de la intrarea standard două valori pentru variabilele C1 și C2, reprezentând valori de capacitati, iar apoi apelează funcții ce calculează capacitatea echivalentă grupării serie, respectiv grupării paralel, după care afișează valorile returnate cu 4 zecimale și aliniere la stânga.
3. Citiți de la tastatură două valori întregi care reprezintă catetele unui triunghi dreptunghic. Determinați ipotenuza și perimetrul triunghiului. Afișați rezultatul pe linii diferite.
4. Citiți de la tastatură două valori întregi a și b ($a \neq 0$), unde a și b sunt coeficienții ecuației $ax+b=0$. Rezolvați ecuația și afișați rezultatul.
5. Considerând că în problema 1 valorile întregi sunt capacități, calculați valorile corespunzătoare grupării serie și paralel. Includeți și situația cu rezistențele și cea cu capacitățile în același program.
6. Se citesc de la tastatură numele a 2 studenți și nota fiecăruia la programare. Să se afișeze pe linii separate numele fiecărui student într-un câmp de afișare cu dimensiunea 20, aliniat la dreapta, respectiv la stânga și media notelor lor, cu o precizie de 2 zecimale.
7. Se citește de la tastatură un număr întreg, ce reprezintă raza unui cerc. Să se afișeze lungimea și aria cercului de rază dată, cu o precizie de 3 zecimale, într-un câmp de afișare cu dimensiunea 10. Pentru valoarea lui PI definiți o constantă simbolică.

8. Se citesc de la tastatură ora plecării unui tren din Cluj și ora la care ajunge la Brașov (oră și minute). Să se afișeze durata călătoriei (în ore și minute).
9. Două mașini se deplasează una spre cealaltă cu vitezele v_1 și v_2 , plecând din două orașe situate la distanța $d=100\text{km}$. Vitezele celor două mașini se citesc de la tastatură. Afișați timpul după care se întâlnesc, exprimat în minute.

4'. Questions

1. What is a data stream?
2. Which are the most important streams in the C language?
3. Which are the most important streams in the C++ language?
4. Which are the main functions that allow I/O operations in the C language?
5. Which are the operators that allow I/O operations in the C++ language?

5'. Homework

1. Implement an application that reads from the standard input 2 values for 2 resistors identified with the R1 and R2 variables. The program calls 2 functions that calculate the series and parallel equivalent resistance. After that, it displays the results right aligned and with 3 digits precision in the fractional part.
2. Implement an application that reads from the standard input 2 values for 2 capacitors identified with the C1 and C2 variables. The program calls 2 functions that calculate the series and parallel equivalent capacity. After that, it displays the results left aligned and with 4 digits precision in the fractional part.
3. Read from the standard input 2 integer values that represent the catheters of a rectangular triangle. Determine the hypotenuse and the perimeter of the triangle. Display the results on different lines.
4. Read from the keyboard 2 integer values, a and b ($a \neq 0$), representing the coefficients of the equation $ax+b=0$. Solve the equation and display the result.
5. Implement the 1-st and 2-nd problems in a single program. Make use of all the possible similarities.
6. Read from the keyboard the names and the marks of 2 students (each student has a single mark). Display, on separate lines, the name of each student (right aligned and left aligned) in two 20 characters fields and their marks with 2 digits precision.
7. Read from the keyboard an integer number that represents the radius of a circle. Display the length and area of the circle with 3 digits precision, in a field that can store 10 characters. Use a symbolic constant for the value of PI.
8. Read from the keyboard a train's exact departure time from Cluj (hour and minutes) and arrival time in Brasov. Calculate and display the travelling time, represented in hours and minutes.
9. Two cars move towards each other, having the speeds v_1 and v_2 and starting from two cities 100km apart. The speeds' values are read from the keyboard. Display how many minutes will pass until the two cars meet in the same point.