FUNCŢII DE INTRARE/IEŞIRE STANDARD

1. Conţinutul lucrării

În lucrare sunt prezentate funcţiile I/E standard, adică funcţiile din biblioteca compilatorului C/C++, care realizează citirea/scrierea din/în fişierele standard de I/E.

2. Consideraţii teoretice

Terminalul standard este terminalul de la care s-a lansat programul. Terminalului standard îi sunt ataşate două fişiere: de intrare (stdin) şi de ieşire (stdout). Ele sunt fişiere secvenţiale.

Funcţiile din biblioteca compilatorului C/C++ utilizate mai frecvent pentru operaţiile de I/E sunt:

- pentru intrare: **getch**, **getche**, **gets**, **scanf**, **sscanf ;**
- pentru ieşire: **putch**, **puts**, **printf**, **sprintf.**

la care se mai adaugă macrourile **getchar** pentru intrare şi **putchar** pentru ieşire.

2.1. Funcţiile getch, getche şi putch

Funcţia **getch** citeşte fără ecou un caracter prin apăsarea unei taste. Tasta poate avea un corespondent ASCII sau o funcţie specială. În primul caz funcţia returnează codul ASCII al caracterului. În al doilea caz, funcţia se apelează de două ori: prima dată returnează valoarea zero, iar a doua oară returnează o valoare specifică tastei acţionate.

Funcţia **getche** este analogă cu funcţia **getch**, realizând însă citirea cu ecou.

Apelul funcţiilor **getch** şi **getche** conduce la aşteptarea apăsării unei taste.

Funcţia **putch** afişează pe ecranul terminalului un caracter corespunzător codului ASCII transmis ca parametru. Caracterele imprimabile au codul ASCII în intervalul [32,126]. Pentru coduri în afara acestui interval se afişează diferite imagini. Funcţia returnează valoarea parametrului de la apel.

Prototipurile acestor trei funcţii se găsesc în fişierul **conio.h** şi sunt:

```
int getch(void);
int getche(void);
int putch(int ch);
```

Exemplu de utilizare:

```
/* Programul  L1Ex1.cpp */

#include <conio.h>
main()
{
putch(getch());
getch();
}
```

2.2. Funcţiile gets *şi* puts

Funcţia **gets** citeşte cu ecou de la terminalul standard un şir de caractere ale codului ASCII, la adresa specificată drept parametru al funcţiei. Din funcţie se revine la:

- citirea caracterului '\n' (newline), caracter care este transformat în caracterul '\0' (null). În acest caz funcţia returnează adresa de început a zonei de memorie în care se păstrează caracterele;
- citirea sfârşitului de fişier (CTRL/Z), funcţia returnând valoarea zero.

Funcţia **puts** afişează la terminalul standard un şir de caractere corespunzând codului ASCII de la adresa transmisă ca parametru. Caracterul '\0' este interpretat ca '\n'. Funcţia returnează codul ultimului caracter afişat sau –1 în caz de eroare.

Prototipurile funcţiilor se găsesc în fişierul **stdio.h** şi sunt:

```
char *gets (char *s);
int  puts (const char *s);
```

Exemplu de utilizare:

```
/*   Programul L1Ex2.cpp  */

#include <stdio.h>
#include <conio.h>
main
{
        char s{200];
        printf("\nIntroduceţi un şir de caractere urmat de
                ENTER\n");
        gets(s);
        printf("\nSirul de caractere introdus\n");
        puts(s);
getch();
        }
```

2.3. Funcţiile scanf şi printf

Funcţia **scanf** are rolul de a introduce date tastate de la terminalul standard sub controlul unor formate. Datele introduse sunt convertite din formatele lor externe în formate interne şi sunt păstrate la adresele specificate la apel. Datele introduse se termină cu apăsarea tastei ENTER.

Prototipul funcţiei **scanf** se găseşte în fişierul **stdio.h** şi este:

```
int scanf(const char *format [,adresa,..]);
```

Ea returnează numărul de câmpuri de la intrare introduse corect sau valoarea EOF(-1) în cazul întâlnirii sfârşitului de fişier (CTRL/Z).

Formatul este specificat ca un şir de caractere. El conţine specificatorii de format, care definesc conversiile din formate externe în formate interne. Un specificator de format este alcătuit din:

- caracterul %;

- opţional caracterul *, care indică faptul că data prezentă la intrare nu se atribuie nici unei variabile;
- opţional un număr zecimal, care defineşte lungimea maximă a câmpului controlat de format;
- 1 sau 2 litere, care definesc tipul conversiei.

Câmpul controlat de format începe cu primul caracter curent care nu este alb şi se termină, după caz:

a) la caracterul după care urmează un caracter alb;
b) la caracterul care nu corespunde tipului de conversie;
c) la caracterul la care se ajunge la lungimea maximă a câmpului.

Datele se citesc efectiv după apăsarea tastei ENTER. Adresa unei variabile se specifică prin **&nume_variabilă.**

Literele care definesc tipul conversiei sunt:

Litera	**Tipul datei citite**
c	char
s	şir de caractere
d	întreg zecimal
o	întreg octal
x, X	întreg hexazecimal
u	unsigned
f	float
ld, lo, lx, lX	long
lu	unsigned long
lf/ Lf	double/long double

Funcţia **printf** este folosită pentru afişarea unor date pe ecranul terminalului standard sub controlul unor formate. Datele sunt convertite din format intern în formatul extern specificat.

Prototipul funcţiei ***printf*** se găseşte în fişierul **stdio.h** şi este:

int printf(const char *format [,expresie, …]);

Formatul este dat ca un şir de caractere. El are în structura sa succesiuni de caractere (care se afişează) şi specificatori de format.

Un specificator de format conţine:

- caracterul %;
- opţional caracterul minus -, care specifică cadrarea datei în stânga câmpului (implicit cadrarea se face în dreapta);
- opţional un număr zecimal, care defineşte dimensiunea minimă a câmpului în care se afişează data;
- opţional un punct urmat de un număr zecimal, care specifică precizia de afişare a datei;
- una sau două litere, care definesc tipul conversiei. Faţă de literele prezentate la scanf apar literele e şi E pentru afişarea datelor float sau double sub formă de exponent, g şi G pentru

afişarea datelor sub forma de exponent sau nu, astfel ca data afişată să ocupe un număr minim de caractere.

Funcţia returnează numărul de caractere (octeţi) afişate la terminal sau –1 în caz de eroare.

Exemple de folosire:

```
/*  Programul L1Ex3.cpp  */

#include <stdio.h>
#include <conio.h>
main( )
{
        int a;
        float b,c;
        printf("\nIntroduceţi o valoare întreagă a=");
        scanf("%5d",&a);
        printf("\nIntroduceţi o valoare reală b=");
        scanf("%5f",&b);
c=a+b;
printf("\nValoarea c=a+b este: %6.3f\n",c);
getch();
}
```

2.4. Funcţiile sscanf şi sprintf

Faţă de funcţiile **scanf** şi **printf**, funcţiile **sscanf** şi **sprintf** au în plus ca prim parametru adresa unei zone de memorie care conţine caractere ASCII. Funcţia **sscanf** citeşte caracterele din această zonă de memorie în loc de zona tampon corespunzătoare fişierului standard de intrare (tastaturii). Funcţia **sprintf** depune caracterele în această zonă de memorie în loc de a fi afişate pe ecran.

Prototipurile acestor funcţii se găsesc în fişierul **stdio.h** şi sunt:

```
int scanf (const char *buffer, const char *format [,adresa, ..]);
int sprintf (char *buffer, const char *format [,adresa, …);
```

Exemplu de folosire:

```
/* Programul L1Ex4.cpp */

#include <stdio.h>
#include <conio.h>
main ( )
{
        char s[100], q[100];
        int a,b;
        float c,d;
        printf ("\nIntroduceti în acelaşi rând valoarea\n\
                lui a şi b despăţite între ele prin blanc\n\
                urmate de ENTER\n");
        gets(s);
```

```
        sscanf(s,"%d %f", &a, &c);
        printf("\n a=%4d  c=%8.3f\n",a,c);
        sprintf(q,"%4d  %8.3f\n",a,c);
        sscanf(q,"%d %f",&b,&d);
        printf("\n b=%5d  d=%9.4f\n",b,d);
        getch();
}
```

2.5. Macrourile getchar şi putchar

Macroul **getchar** permite citirea cu ecou a caracterelor codului ASCII, deci nu a celor corespunzătoare tastelor speciale. Caracterele tastate se introduc într-o zonă tampon până la acţionarea tastei ENTER. La revenire, se returnează codul ASCII al primului caracter introdus, iar la un nou apel, al următorului caracter introdus ş.a.m.d. La întâlnirea sfârşitului de fişier (CTRL/Z) se returnează valoare EOF(-1).

Macroul **putchar** afişează caracterul al cărui cod ASCII s-a transmis.

Macrourile **getchar** şi **putchar** sunt definite în fişierul **stdio.h** şi au formatele:

```
int getchar(void);
int putchar (int c);
```

şi se apelează exact ca funcţiile **getch** şi **putch**.

Exemplu de utilizare:

```
/*  Programul L1Ex5.cpp  */

#include <stdio.h>
#include <conio.h>
main( )
{
        putchar(getchar)() ;
        putchar('\n');
        getch();
}
```

Funcțiile de intrare-ieșire CIN şi COUT

Output cu COUT

Am văzut că pentru a putea folosi obiectele `cout` (console output) şi `cin` avem nevoie de *biblioteca* (library) `iostream`. Ca să putem folosi această bibliotecă trebuie să includem următoarele linii la începutul programului:

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

Afişarea numerelor sau stringurilor pe ecran (consolă) se face cu obiectul `cout` şi operatorul de inserţie <<. De exemplu:

```
cout << "Quick wafting zephyrs vex bold Jim\n"
     << "The five boxing wizards jump quickly.\n";
```

`cout` afişează datele aşa cum le daţi. cout nu formatează nimic, nu adaugă spaţii între cuvinte, nu adaugă new line, etc. Exemplu:

```
cout << "Quick" << "wafting" << "zephyrs";
```

Se va afişa:
`Quickwaftingzephyrs`

Dacă vrem spaţii între cuvinte atunci adăugăm şi spaţii:

```
cout << "Quick" << " " << "wafting" << " " << "zephyrs";
```

Lanţul cout poate fi oricât de lung vreţi. Nu e obligatoriu să-l aveţi pe un singur rând (vezi primul exemplu). Trebuie să se termine cu punct şi virgulă. Puteţi, de asemenea, să aveţi expresii într-o instrucţiune cout:

```
cout << "Aria cercului este:" << (PI * raza * raza);
```

Practic orice obiect care are o reprezentare string poate fi afişat pe ecran cu cout. Aţi văzut că un rând nou se inserează cu secvenţa escape `'\n'`. Ei bine, mai este o metodă cu `endl`. De exemplu:

```
cout << "Quick wafting zephyrs vex bold Jim" << endl << "The
five boxing wizards jump quickly.\n";
```

Atunci când vreţi să afişaţi numere `double` s-ar putea să nu obţineţi ceea ce vreţi.

```
double phi = 4.893654;
cout << phi;
```

Se va afişa `4.893654`. Dar poate vreţi să afişaţi doar primele două zecimale. Cum faceţi asta? Cu următoarele instrucţiuni:

```
cout.setf(ios::fixed);
cout.setf(ios::showpoint);
cout.precision(2);
```

Prima instrucţiune ne permite să folosim funcţia `precision` **doar** pentru partea fracţionară (de după punct); altfel ar fi luat în considerare tot numărul.
A doua instrucţiune afişează punctul zecimal de fiecare dată - chiar şi pentru numere întregi.
A treia instrucţiune setează precizia numărului la 2 zecimale. Se fac rotunjiri! Argumentul funcţiei trebuie să fie pozitiv şi număr întreg sau o expresie evaluată la `int`.
După aceste instrucţiuni puteţi folosi cout normal ca să afişaţi numerele reale în noul format:

```
cout << phi; // 4.89
```

Puteţi folosi opţiunea `ios::scientific` ca să afişaţi în notaţie ştiinţifică:

```
double phi = 0.0000123;
cout.setf(ios::scientific);
cout.setf(ios::showpoint);
cout.precision(2);
cout << phi; // 1.23e-005
```

Input cu CIN

Similar putem folosi `cin` (console input) pentru operaţii de input, adică de obţinere a datelor de la tastatură (de la user). Se foloseşte cu operatorul de extracţie `>>`.

```
int a, b;
cout << "Introduceti doua numere: ";
cin >> a >> b;
cout << "Suma lor este: " << (a + b);
```

Atunci când întâlneşte instrucţiunea `cin`, programul aşteaptă inputul de la user. Atribuie prima valoare primei variabile, a doua valoare variabilei a doua, etc. Programul nu citeşte datele de intrare decât după ce utilizatorul apasă ENTER la tastatură. În acest fel userul se poate corecta folosind backspace. Obiectul `cin` foloseşte **spaţiile albe** (space, enter, tab, etc.) ca delimitatoare. Asta înseamnă că datele de intrare trebuie despărţite prin câte un spaţiu sau rând nou (new line). `cin` ignoră - şi elimină din fluxul (stream) de intrare - toate spaţiile albe până întâlneşte un input valid.

Deoarece ignoră spaţiile albe, nu puteţi citi propoziţii de cuvinte cu cin. Trebuie să folosiţi funcţia `getline` (care citeşte până când întâlneşte un caracter new line pe care îl extrage din stream şi-l ignoră). Se foloseşte cu tipul `string`:

```
string fullname;
getline(cin, fullname);
cout << "\nNumele tau este: \n"
     << fullname << endl;
```

Primul parametru trebuie să fie `cin` pentru că citiţi de la tastatură. Al doilea parametru este o variabilă string în care va fi salvat şirul. Streamul (fluxul) de intrare (input stream) reprezintă şirul datelor de intrare. Imaginaţi-vă un flux de informaţii care *curge* către calculator. Aceste date vin nu numai de la tastatură, ci şi de la alte dispozitive de intrare: mouse, scanner, mircofon, etc. `cin` se ocupă numai de tastatură.

Similar, `cout` scrie în streamul de ieşire (output stream) care este afişat pe ecran. Exemplu: Se citesc de la tastatură trei numere naturale. Să se afişeze suma lor.
Pentru datele de intrare: `14 89 99` se va afişa `202`.

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

int main()
{
    int a, b, c;
    cin >> a >> b >> c;
    cout << a + b + c;  // parantezele pot lipsi

    system("PAUSE");
    return 0;
}
```

Cu cin puteţi citi numere întregi şi reale, stringuri şi caractere, etc.

3. Mersul lucrării

3.1. Se vor executa programele date ca exemplu în lucrare şi se vor analiza rezultatele obţinute.
3.2. Scrieţi un program pentru a verifica modul de execuţie a funcţiei getch când se apasă o tastă care corespunde unei funcţii speciale.
3.3. Scrieţi un program pentru a verifica ce se afişează de către funcţia putch atunci când parametrul său este o valoare în afara intervalului [32,126].
3.4. Scrieţi un program care afişează codurile ASCII ale caracterelor corespunzătoare tastaturii.
3.5. Scrieţi un program care afişează caracterele corespunzătoare codurilor ASCII din intervalul [32,126].
3.6. Scrieţi un program care să conţină apelul gets(s), unde s a fost definit ca un tablou.
Verificaţi ce conţine fiecare element al tabloului. De ce caracterul '\n' a fost înlocuit cu '\0'?

3.7. Scrieţi un program care citeşte un şir de litere mici şi le afişează sub formă de litere mari.
3.8. Scrieţi un program care citeşte un şir de litere mari şi le afişează sub formă de litere mici.
3.9. Scrieţi un program care realizează suma, diferenţa, produsul şi împărţirea a două numere reale. Afişarea se va face sub formă tabelară:

x	y	x + y	x – y	x * y	x / y

3.10. Scrieţi un program pentru a verifica modul de afişare a valorii lui π = 3.14159265 cu diferiţi descriptori de format.
3.11. Scrieţi un program pentru afişarea unui întreg citit de la tastatură în octal şi hexazecimal.