INSTRUCŢIUNEA DE ATRIBUIRE. OPERATORI ARITMETICI

1. SCOPUL LUCRĂRII

În această lucrare se vor studia următoarele:
-Funcţii de citire / scriere:
-Instrucţiunea de atribuire
-Operatori aritmetici

2. BREVIAR TEORETIC

2.1. Funcţii de citire / scriere

Orice program are anumite date de intrare, pe care le prelucrează şi în final afişează rezultatul. Datele de intrare, cât şi rezultatele parţiale şi finale, sunt stocate (memorate) în variabile.

Datele de intrare ale programului pot fi:
- citite de la tastatura (tastatura este principala sursă de date de intrare)
- iniţializate direct prin instrucţiunea de atribuire
- iniţializate cu numere generate aleator
- citite dintr-un fişier de pe disc
- recepţionate dintr-un port de intrare al calculatorului (de exemplu de pe portul serial sau paralel al PC-ului)

Astfel, funcţia **scanf()** este folosită pentru a citi date de la tastatură sub controlul unui format de citire. Datele citite sunt memorate în variabilele specificate prin argumentele funcţiei.

Modul de utilizare tipic:
scanf(şirul de formate, adresele variabilelor în care se citesc datele);
Formatele de citire cele mai folosite sunt:

%d	pentru tipul de date int
%c	pentru tipul de date char
%lf	pentru tipul de date double (Atenţie: este **lf** şi nu **1f**)

Exemplu:

Pentru a citi un număr întreg de la tastatură pe care să-l memorăm în variabila număr, se foloseşte secvenţa de program:

```
int numar;
scanf("%d",&numar);
```

Exemplu:

Pentru a citi de la tastatură, folosind o singură instrucţiune **scanf(),** 4 valori numerice, şi anume: două numere întregi, un caracter şi un număr real, se foloseşte secvenţa de program:

```
int i1,i2;
char c; double r;
scanf("%d%d%c%lf",&i1,&i2,&c,&r);
```

Datele de citit, se tastează separate prin caracterele: spaţiu (blank) sau TAB sau ENTER.

Pentru a citi date de tip caracter se poate folosi şi instrucţiunea **getch()**. Această funcţie returnează codul ASCII al tastei apăsate.

Exemplu:

```
int c=getch();
```

Folosind funcţia **getch()**, caracterului tastat nu i se face ecou pe ecran. Pentru a i se face ecou, se foloseşte o variantă a acestei funcţii, şi anume funcţia **getche().**

Atât funcţia **getch(),** cât şi funcţia **scanf(),** aşteaptă până când utilizatorul tastează datele.

Pentru a detecta dacă s-a apăsat sau nu o tastă, fără a aştepta apăsarea efectivă, se foloseşte funcţia **kbhit()**. Aceasta funcţie nu are nici un parametru. Ea întoarce ca rezultat, valoarea 0, dacă nu a fost apăsată nici o tastă, şi o valoare diferită de 0, dacă a fost apăsată o tastă. Funcţia **kbhit()**, examinează bufferul tastaturii în care sunt automat stocate caracterele tastate. Dacă nu este nici un caracter prezent, întoarce drept rezultat valoarea 0. Funcţia **kbhit()** nu aşteaptă apăsarea unei taste, programul continuând cu următoarea instrucţiune din program.

Exemplu:

```
int este=kbhit();
```

Pentru a afişa date pe monitor, în modul text, se foloseşte funcţia **printf()**. Ea afişează texte şi eventual, conţinutul unor variabile, sub controlul unor formate de afişare, începând de la poziţia curentă a cursorului. În cazul textului, pot apărea anumite caractere speciale, ce nu sunt afişate ca atare, denumite *secvenţe escape.*

Caractere speciale uzual folosite în textul de afişat sunt:

\n - are rolul de a muta cursorul la începutul liniei următoare

\t - are rolul de a avansa cursorul cu un TAB.

\\ - afişează caracterul \
\" - afişează caracterul "

Exemplu:
printf("Numele directorului curent este: C:\\TURBO");

Pentru a afişa, pe lângă texte, şi conţinutul unor variabile, sintaxa, în cazul general, este următoarea:
printf("Sirul de texte si formate",lista de variabile);

Cele mai utilizate formate de afişare, sunt următoarele:
%d pentru variabile de tipul int
%c pentru variabile de tipul char
%lf pentru variabile de tipul double. (Atenţie: este **lf** şi nu **1f**)
%f pentru variabile de tipul float

Exemplu:
printf("x=%d,y=%d",x,y);

2.2. Instrucţiunea de atribuire

Valoarea depozitată într-o locaţie de memorie, poate fi schimbată printr-o operaţie de atribuire. Ca operator de atribuire, în limbajul C, se foloseşte semnul egal.
Sintaxa operaţiei de atribuire este următoarea:

nume_variabilă = expresie ;

Exemplu:

x=2.5;

Se citeşte: x "primeşte" valoarea 2.5.
În urma acestei operaţii, în variabila x se memorează valoarea 2.5 . Variabila x trebuie declarată de tipul real (float sau double).
Exemplu:

x=x+1;

Se citeşte: x "primeşte" vechea valoare a lui x crescută cu 1.

Această operaţie de creştere a conţinutului unei variabile cu o unitate, se cheamă **incrementare**. Incrementarea fiind o operaţie des întâlnită în programe, în limbajul C există un operator special ce o realizează: operatorul ++ . Acesta este un operator unar şi el poate fi plasat fie la stânga, fie la dreapta unui operand. Acţiunea lui va fi diferită însă, depinzând de poziţia lui (în stânga sau în dreapta operandului).

Exemplu:

```
int x,y;
x=7;
y=x++;
```

Operatorul **++**, în acest exemplu, este plasat la dreapta unui operand: întâi se va face operaţia de atribuire (y=x) şi apoi se va face incrementarea variabilei x (x=x+1). Astfel, în final se va obţine y=7 şi x=8.

Exemplu:

```
int x,y;
x=7;
y=++x;
```

Operatorul **++**, în acest exemplu, este plasat la stânga unui operand: întâi se va face incrementarea variabilei x (x=x+1) şi apoi se va face operaţia de atribuire (y=x). Astfel, în final se va obţine y=8 şi x=8.

Pentru **decrementarea** unei variabile (scăderea valorii variabilei cu o unitate), există în limbajul C un operator special: operatorul **--** . Acest operator se foloseşte la fel ca operatorul de incrementare.

2.3 Operatori aritmetici

Sunt 5 operatori aritmetici de bază, şi anume:

+	adunare
-	scădere
*****	înmulţire
/	împărţire
%	restul împărţirii (modulo)

Operatorul de împărţire **/** , în cazul numerelor întregi, calculează trunchind rezultatul exact.

Astfel 7 / 2 este evaluat ca 3 si nu ca 3.5

Toţi aceşti operatori sunt operatori binari (adică cu operanzi şi în stânga şi în dreapta).

Operatorul **%** poate avea drept operanzi doar numere întregi.

Operatorii aritmetici, au priorităţile cunoscute de la aritmetică. Astfel, în expresia *a+b*c* întâi se calculează produsul şi apoi suma.

Observaţie importantă:

Deoarece limbajul C are foarte mulţi operatori, a căror prioritate nu mai este chiar aşa de evidentă ca la cei aritmetici, se recomandă, atât pentru claritatea programului, cât şi pentru evitarea

greşelilor, folosirea parantezelor. Totdeauna operaţiile cuprinse între paranteze se execută primele.

3. DESFĂŞURAREA LUCRĂRII

Se vor edita şi apoi executa programele descrise în continuare.

Programul nr. 1

Sa se afişeze următorul text, citat din Biblie:

"Cautati mai intai Imparatia lui Dumnezeu !"

Se vor afişa inclusiv ghilimelele.

Sursa programului:

```
#include<stdio.h>
#include<conio.h>
void main(void)
{
 clrscr();
 printf("\"Cautati mai intai Imparatia lui Dumnezeu !\"");
 getch();
}
```

Programul nr. 2

Să se comute conţinutul a două variabile a şi b, între ele.

Algoritm:

-se salvează în variabila auxiliară aux, variabila a
-se copiază în a variabila b
-se copiază în b variabila aux

Greşeli frecvente:

-Nu se foloseşte o variabila auxiliară, ci se face comutarea direct: a=b şi apoi b=a.

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int a, b, aux;
 clrscr();
 printf("a="); scanf("%d",&a);
 printf("b="); scanf("%d",&b);
 aux=a;
 a=b;
```

```
 b=aux;
 printf("Dupa comutare: a=%d, b=%d",a,b);
 getch();
}
```

Programul nr. 3

Calculul mediei aritmetice a 3 numere întregi, citite de la tastatură.

Algoritm:

- se calculează suma numerelor, şi rezultatul se împarte la 3.

Greşeli frecvente:

- se declară variabila medie, de tip întreg
- se calculează medie=(a+b+c)/3 , ceea ce este greşit căci rezultatul se trunchiază la întreg.

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int a,b,c;
 double medie;
 clrscr();
 printf("a="); scanf("%d",&a);
 printf("b="); scanf("%d",&b);
 printf("c="); scanf("%d",&c);
 medie=(a+b+c)/3.0;
 printf("medie=%lf",medie);
 getch();
}
```

int (void) #include<iostream> using namespace std; cout<<"a= "; cin>>a; return 0;

Programul nr. 4

Se citesc trei numere întregi a, b şi c de la tastatură, coeficienţi ai polinomului de gradul doi $P(x)=ax^2+bx+c$. Să se calculeze şi afişeze valoarea polinomului pentru o valoare particulară întreagă x, citită de la tastatură.

Sursa programului:

```
# include <conio.h>
# include <stdio.h>
void main (void)
{
 int a,b,c,x;
```

```
 int P; //presupunem ca nu se depaseste capacitatea de stocare a
//tipului int
 printf("Tastati 3 numere intregi, a, b si c: ");
 scanf("%d%d%d",&a,&b,&c);
 printf("x="); scanf("%d",&x);
 P=a*x*x+b*x+c;
 printf ("P=%d",P);
}
```

Programul nr. 5

Se citeşte un număr întreg de la tastatură. Acest număr are semnificaţia unui interval de secunde. Să se afişeze intervalul în ore, minute şi secunde.

Exemplu:

Dacă numărul citit este 61, trebuie să se afişeze:
Nr. Ore = 0; Nr. Minute = 1; Nr. Secunde = 1

Sursa programului:

```
#include <stdio.h>
void main(void)
{
 int N; // intervalul in secunde citit
 int nrOre; int nrMin; int nrSec;
 printf("Tastati durata in secunde a intervalului: ");
 scanf("%d",&N);
 nrOre = N/3600;
 nrMin = (N % 3600)/60;
 nrSec = N - nrOre * 3600 - nrMin * 60;
 printf("\nNr. Ore=%d",nrOre);
 printf("\nNr. Minute=%d",nrMin);
 printf("\nNr. Secunde=%d",nrSec);
}
```

Programul nr. 6

Se citeşte un număr întreg, pozitiv, de la tastatură. Să se calculeze cifra unităţilor, cifra zecilor şi cifra sutelor.

Algoritm:

- cifra unităţilor este restul împărţirii numărului la 10
- pentru calcul cifrei zecilor, se împarte numărul iniţial la 10. În noul număr obţinut, vechea cifră a zecilor a devenit cifra unităţilor. Aceasta se află ca restul împărţirii noului număr la 10.

- cifra sutelor se obţine dupa acelaşi algoritm ca şi cifra zecilor.

Greşeli frecvente:

- se împarte direct numărul la 10, pentru a afla cifra zecilor
- se imparte direct numărul la 100, pentru a afla cifra sutelor

Sursa programului:

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int nr;
 int u,z,s;//unitati, zeci, sute
 clrscr();
 printf("nr=");
 scanf("%d",&nr);
 //cifra unitatilor:
 u=nr%10;
 //calcul cifra zecilor:
 nr=nr/10;
 z=nr%10;
 //cifra sutelor:
 nr=nr/10;
 s=nr%10;
 //afisare rezultate:
 printf("u=%d, z=%d, s=%d",u,z,s);
 getch();
}
```

Programul nr. 7

Se citeşte un număr natural de la tastatură, care constă din cel puţin două cifre. Să se calculeze noul număr obţinut prin comutarea cifrei unităţilor cu cifra zecilor. Exemplu:

Pentru numărul iniţial 4532, după comutare se obţine 4523.

Sursa programului:

```
# include <stdio.h>
void main (void)
{
 int nr;
 printf("Dati un numar natural >= 10: ");
 scanf("%d",&nr);
 int u;//cifra unitatilor
```

```
 u=nr%10;
 int z;//cifra zecilor
 z=(nr/10)%10;
 int ramas;//numarul ramas prin eliminarea cifrei zecilor si  a cifrei
//unitatilor
 ramas=nr/100;
 //Dupa comutare, numarul devine:
 nr=ramas*100+10*u+z;
 printf("Dupa comutare, nr=%d",nr);
}
```

Programul nr. 8

Se citesc doua numere naturale a si b, capetele unui interval inchis. Sa se genereze un numar aleatoriu in intervalul [a, b]. Sa se afiseze acest numar.

Sursa programului:

```
#include<stdio.h>
#include<stdlib.h>
void main(void)
{
 int a,b,nr;
 printf("a=");
 scanf("%d",&a);
 printf("b=");
 scanf("%d",&b);
 randomize();
 nr=a+random(b-a+1);
 printf("Numar=%d",nr);
}
```

```
#include<iostream>
#include<time.h>
int
cout<<"a= ";  cin>>a;
srand(time(NULL));
nr=a+rand()%(b-a+1);
return 0;
cout<<"Numar = "<<nr;
```

4. PROBLEME PROPUSE

1.Sa se afişeze un dreptunghi ale cărui linii sunt formate din steluţe (caracterul *)

2. Se citeşte un număr natural care are cel puţin trei cifre. Să se calculeze noul număr obţinut prin comutarea cifrei unităţilor cu cifra sutelor.

3. Sa se genereze şi afişeze trei numere întregi, aleatoare în intervalul închis [10,100].