Programare procedurală
- Laborator 2 -

Instrucțiuni ale limbajului C

1. **Instrucțiunea vidă**

```
;
```

2. **if-else**

```
if(expresie)
{
        set de instrucțiuni
}
else
{
        set de instrucțiuni
}
```

Observații:

- ***else*** – opțional;
- se pot imbrica;

3. **switch**

```
switch(expresie)
{
   case valoare_1:
            set de instrucțiuni
            break;
    case valoare_2:
            set de instrucțiuni
            break;
   ……….
   default:
            set de instrucțiuni
            break;
}
```

Observații:

- **valoare_i** este o valoare posibilă a expresiei din switch;
- absența instrucțiunii ***break;*** determină continuarea la următoarea instrucțiune.
- ***default***: opțional.

4. **while**

```
while(expresie)
{
      set de instrucțiuni;
}
```

5. **do… while**

```
do
{
        set de instrucțiuni;
} while(expresie);
```

6. **for**

```
for(expresie1;expresie2;expresie3)
{
        set de instrucțiuni;
}
```

Exemplu:

```
int i,j;
for(i=1,j=23;i<=10 && j>=1;i++,j-=2) //C99 admite și for(int i=1,j=23;i<=10 && j>=1;i++,j- =2)
{ …… }
```

Instrucțiuni speciale:

7. **break** - determină ieșirea forțată dintr-o structură repetitivă.
Exemplu:

```
int i,a,j=0;
scanf("%d",&a);
for(i=0; ;j++)
{
      if(i>a)
            { printf("%d ",j-1);
              break;
            }
      i+=2;
}
```

8. **continue –** folosit în instrucțiunile repetitive; determină încheierea iterației curente și saltul la iterația următoare.

Exemplu:

```
int i=0,a;
scanf("%d",&a);

while(i<=a)
{
   i++;
   if(i%2==0)
         continue;
   printf("%d ",i);
}
```

9. **goto** – salt la un set de instrucțiuni etichetate (**nerecomandată**, programe greu de urmărit**)**.

Exemplu:

```
int i,j,k,a;
scanf("%d",&a);

eticheta1:
{
   i=0; j=0;
for(;;)
   {
      i+=2;
      if(i>a)
         goto eticheta2;
      j++;
   }
}
eticheta2:
   {
      printf("Catul impartirii la 2 este: %d!\n Introduceti alt numar pozitiv: ",j);
      scanf("%d",&a);
      if(a>0)
         goto eticheta1;
   }
```

10. return

```
int d,x;
scanf("%x",&x);
if(x%2==0 && x!=2)
{
   printf("Neprim!");
   return 0;
}
for(d=3;d<=sqrt(x);d+=2)
{
   if(x%d==0)
      {
         printf("Neprim!");
         return 0;
      }
}
printf("Prim!");
```

Util (pentru *operații pe biți*)**:**
http://campion.edu.ro/arhiva/www/arhiva_2009/papers/paper21.pdf

Probleme

Observație: Problemele 1-3 se vor rezolva folosind operații pe biți.

1. Pentru valori întregi citite de la tastatură să se tiparească valoarea corespunzătoare în binar.

Pentru verificare:
system("C:\\windows\\system32\\calc.exe");

2. Se citesc 2 numere întregi x și n unde n este între 0 si 15. Să se afișeze: bitul n din x, numărul x în care se seteaza bitul n, numărul x în care se șterge bitul n, numărul x în care se complementează bitul n.

3. Se citesc întregii x, y, n, p. Să se copieze în x, începând din poziția p, ultimii n biți din y și să se afișeze noua valoare a lui x.

4. Scrieți un program care primește ca input de la tastatură scrierea unui număr în baza 2 și calculează direct scrierea acestuia în baza 16 (nu mai trece prin baza intermediară 10). Realizați acest lucru inversând cele două baze (input – scrierea în baza 16, output – scrierea în baza 2).

5. Se citesc numere naturale până la întâlnirea numărului 0. Să se afișeze toate perechile de numere consecutive citite cu proprietatea că al doilea număr reprezintă restul împărțirii primului număr la suma cifrelor sale.

6. Se citește de la tastatură un număr natural p. Să se determine toate perechile distincte de numere întregi (i,j,k) cu proprietatea că ele pot reprezenta laturile unui triunghi de perimetru p. Folosiți maxim două instrucțiuni for.