Programare in C - Curs 3

break

- Instructiune care forteaza iesirea din instructiunea repetitiva curenta
- Utilizata atunci cand este nevoie sa parasim o bucla fara a astepta repetarea testului (initial sau final)
- Daca sunt mai multe bucle imbricate break forteaza iesirea doar din ultima bucla deschisa

Exemplu break

```
for(i=0;;i++)
{
        if(i%10==0)
                break;
....
}
```

```
for(i=0;i<20;i++)
{
        for(j=0;j<20;j++)
        {       //forteaza iesirea din al doilea
                //for cand i==j si i este par
                if(j==i && i%2==0)
                        break;
        }
        //dupa break se continua executia
        //primului for de aici
        …
}
```

switch

```
switch (expresie)
    {
    declaratii
    case expresie_constanta:
        instructiuni
        break;
case alta_expresie_constanta:
        instructiuni
        break;
default:
        instructiuni

}
```

Exemplu switch

```
Alegeti una din urmatoarele optiuni:
S - Start program
O - Optiuni program
X - Iesire program
s
s-a executat programul
Alegeti una din urmatoarele optiuni:
S - Start program
O - Optiuni program
X - Iesire program
o
nici o optiune disponibila
Alegeti una din urmatoarele optiuni:
S - Start program
O - Optiuni program
X - Iesire program
S
s-a executat programul
Alegeti una din urmatoarele optiuni:
S - Start program
O - Optiuni program
X - Iesire program
O
nici o optiune disponibila
Alegeti una din urmatoarele optiuni:
S - Start program
O - Optiuni program
X - Iesire program
x
```

```c
char c;
int exit=0,print=1;
while(1)
{
        if(print)
        {
                printf("Alegeti una din urmatoarele optiuni:\n");
                printf("S - Start program\n");
                printf("O - Optiuni program\n");
                printf("X - Iesire program\n");
        }
        c=getchar();
        switch(c)
        {
                case 's':
                case 'S': printf("s-a executat programul\n"); print=1;break;
                case 'o':
                case 'O': printf("nici o optiune disponibila\n");print=1; break;
                case 'x':
                case 'X': exit=1; break;
                default:print=0;

        }
        if(exit==1)
                break;

}
```

switch

- expresiile constante dupa case = trebuie cunoscute la compilare.
  - nu pot fi folosite variabile (care n-au fost declarate de tip const)
- dupa ce s-a intrat pe un “case”, daca nu exista break se executa toate instructiunile urmatoare
- expresiile constante trebuie sa fie int, char si nu au voie sa se repete in switch

exemplu switch fara break

```
Alegeti una din urmatoarele optiuni:
S - Start program
O - Optiuni program
X - Iesire program
s
s-a executat programul
nici o optiune disponibila
```

```
char c;
int exit=0,print=1;
while(1)
{
        if(print)
        {
                printf("Alegeti una din urmatoarele optiuni:\n");
                printf("S - Start program\n");
                printf("O - Optiuni program\n");
                printf("X - Iesire program\n");
        }
        c=getchar();
        switch(c)
        {
                case 's':
                case 'S': printf("s-a executat programul\n"); print=1;
                case 'o':
                case 'O': printf("nici o optiune disponibila\n");print=1;
                case 'x':
                case 'X': exit=1; break;
                default:print=0;

        }
        if(exit==1)
                break;

}
```

continue

- forteaza trecerea la urmatoarea iteratie din bucla curenta
- in while si do - trece direct la test (atentie ca nu se face actualizare automata)
- in for se face actualizare si apoi se trece la test

```
int i=0;
for(;i<10;i++)
{
        if(i%3==0)
                continue;
        printf("i=%d\n",i);
}
```

Exemplu continue

```
int i=0;
for(;i<10;i++)
{
        if(i%3==0)
                continue;
        printf("i=%d\n",i);
}
```

```
[@web316:W,06:44 AM,Sun Oct 20]>./a.out
i=1
i=2
i=4
i=5
i=7
i=8
```

Exemplu continue

```
int i=0;
for(;i<10;i++)
{
        if(i%3==0)
                continue;
        printf("i=%d\n",i);
}
```

```
[@web316:W,06:44 AM,Sun Oct 20]>./a.out
i=1
i=2
i=4
i=5
i=7
i=8
```

- asa nu
  - i++ nu se mai executa dupa ce se intalneste primul nr divizibil cu 3
  - se intra in ciclu infinit

```
int i=1;
while(i<10)
{
        if(i%3==0)
                continue;
        printf("i=%d\n",i);
        i++;
}
```

```
[@web316:W,06:49 AM,Sun Oct 20]>./a.out
i=1
i=2
^C
[@web316:W,06:50 AM,Sun Oct 20]>
```

Break & continue

- Dezbateri aprinse despre când ar trebui utilizate si cand nu
  - Este preferabil sa scrieti conditia initiala astfel incat sa nu fie necesar break
  - daca totusi folositi break sa-l folositi dintr-un if.
  - Aveti grija sa fie comentata zona astfel incat sa se inteleaga de ce se iese din ciclul respectiv

goto

- nerecomandat, nu se va folosi la curs, laborator, teme
- sintaxa:
  - goto eticheta
- eticheta se defineste
  - eticheta: instructiune
- exemplu

```
        int i=0;
eticheta:
        if(i%3!=0)
                printf("i=%d\n",i);
        i++;
        if(i<10)
                goto eticheta;
        return 0;
```

```
[@web316:W,07:15 AM,Sun Oct 20]>./a.out
i=1
i=2
i=4
i=5
i=7
i=8
```

Functia in C

- set de instructiuni ce efectueaza o sarcina specifica
- caracteristicile unei functii
  - poate primi parametri
  - modul in care este implementata poate fi ascuns
  - poate intoarce rezultate
  - rezultatele pot fi folosite sau pot fi ignorate

Sintaxa definirii unei functii

- tip_rezultat nume_functie([tip_parametru1 nume_parametru1][,tip_parametru_i nume_parametru_i){//declaratii variabile, instructiuni}

Exemplul 1

```
int min(int a, int b)
{
        return a<b?a:b;
}
```

- intoarce rezultat de tip intreg
- are numele min
- are 2 parametri: a si b
- a<b?a:b intoarce a daca a<b sau b daca b<=a
- return expresie - evalueaza expresia si intoarce rezultatul acesteia ca rezultat al functiei

Prototipul unei functii

tip_rezultat nume_functie([tip_parametru1 nume_parametru1][,tip_parametru_i nume_parametru_i)**;**

Ex:

int min(int a, int b);

- se foloseste pentru a preciza ca o functie (cu numele min, cu 2 parametri de tip intreg si care intoarce un rezultat intreg) va fi definita mai jos in program

Domenii de vizibilitate ale variabilelor (Scope)

- Variabile globale
  - Definite in afara functiilor
  - Pot fi folosite in interiorul oricarei functii
  - Nerecomandate
    - Pot fi modificate de oriunde din program
    - => dificil de urmarit, pot fi introduse erori dificil de identificat
    - Cu cat programul este mai mare cu atat este mai greu de controlat unde se modifica o variabila
- Variabile locale (nivel de bloc, inclusiv functie)
  - Variabila este vizibila in blocul declarat si in toate blocurile interne (imbricate, “*nested blocks*”)

Exemplu

```
int varTest=0;

int f1()
{
        varTest+=1;
        return varTest;
}

int main()
{
        int varTest=5;
        printf("varTest=%d\n",varTest);
        f1();
        printf("varTest=%d\n",varTest);
        return 0;


}
```

```
varTest=5
varTest=5
```

```
int varTest=0;

int f1()
{
        varTest+=1;
        return varTest;
}

int main()
{
        int var=5;
        printf("varTest=%d, var=%d\n",varTest,var);
        var=f1();
        printf("varTest=%d, var=%d\n",varTest, var);
        return 0;


}
```

```
varTest=0, var=5
varTest=1, var=1
```

Accesibilitatea variabilelor

- Daca avem 2 variabile cu acelasi nume declarate in blocuri imbricate variabila declarata in blocul interior o va ”ascunde” pe cea din blocul exterior

Domenii de vizibilitate ale variabilelor(2)

- O variabila poate fi declarata in orice bloc de instructiuni si este vizibila strict in acel bloc

```c
int varTest=0;

int f1()
{
        varTest+=1;
        return varTest;
}

int main()
{
        int varTest=5;
        printf("varTest=%d\n",varTest);
        f1();
        if(1)
        {
                float varTest=1;
                printf("varTest=%f\n",varTest);
        }
        printf("varTest=%d\n",varTest);
        return 0;
```

```
varTest=5
varTest=1.000000
varTest=5
```

```c
/*int varTest=0;

int f1()
{
        varTest+=1;
        return varTest;
}
*/
int main()
{
//      int varTest=5;
//      printf("varTest=%d\n",varTest);
//      f1();
        if(1)
        {
                float varTest=1;
                printf("varTest=%f\n",varTest);
        }
        printf("varTest=%d\n",varTest);
        return 0;
```

```
c3p5varScope.c: In function 'main':
c3p5varScope.c:21: error: 'varTest' undeclared (first use in this function)
c3p5varScope.c:21: error: (Each undeclared identifier is reported only once
c3p5varScope.c:21: error: for each function it appears in.)
```

Durata de viață a variabilelor

- Static
  - Variabile declarate cu cuvantul cheie static
  - Durata de viata = durata de viata a programului
  - Pot exista si sa nu fie vizibile
  - Detalii aici: http://stackoverflow.com/questions/572547/what-does-static-mean-in-a-c-program
- Automatic
  - Exista pe perioada blocului in care au fost definite
- Dinamic
  - Folosită la alocarea dinamica