Utasítások, elágazás- és ciklusszervezés

- ***C nyelvi utasítások***
  - *Kifejezés utasítás*
  - *Üres utasítás*
  - *Összetett, vagy blokk-utasítás*
  - *Elágazásszervező utasítások*
    - ***Az* if *utasítás***
    - ***Az* if else *szerkezet***
    - ***A* switch *többirányú elágaztató utasítás***
    - ***A* goto *utasítás***
  - *Ciklusszervező utasítások*
    - ***A* while *ciklusszervező utasítás***
    - ***A* for *ciklusutasítás***
    - ***A* do while *ciklus***
  - *Ciklusleftás módosítása:* break, continue *utasítások*

C nyelvi utasítások

A C nyelvben is a

program = adatszerkezetek + algoritmusok

képlet igaz, ahol az algoritmusokat utasítások sorozatával adjuk meg.

Kifejezés utasítás

Bármelyik kifejezésből utasítás lesz, ha pontosvesszőt teszünk utána.

Pl.: *valt1 = a +25 ;* /* *értékadó utasítás* */

Üres utasítás: *csak pontosvesszőt tartalmaz. Formailag utasítást igénylő helyre írjuk, ha ott egyébként nem kell semmit végrehajtani. Gyakran előfordul a* **while**, **do**, **for** *utasításokban.*

Pl.: **do ; while** (!kbhit()); /* billentyűnyomásig vár */

C nyelvi utasítások . .

Összetett, vagy blokk-utasítás

```
{
    <deklarációk és definíciók>
    <utasítások>
}
```

Mind a deklarációk, definíciók, mind az utasítások elmaradhatnak.

Összetett utasítás alkalmazandó olyan helyen, ahol formailag csak egy utasítás állhat, de több utasítást akarunk elvégeztetni.

Láthatóság: *Az összetett utasításban deklarált ill. definiált objektumok csak a blokkon belül láthatók, hivatkozhatók.*

C nyelvi utasítások . .

Elágazásszervező utasítások:

if, if else, switch, goto

Az if utasítás *alkalmas egy feltételtől függően egy programrész végrehajtására, vagy átugrására. Alakja:*

if (<kifejezés> **)**
<utasítás>

Ha a <kifejezés> *igaz (értéke nem nulla), akkor végrehajtódik az* <utasítás>*, egyébként az* <utasítás> *utáni programrészen folytatódik a program futása.*

Elágazásszervező utasítások

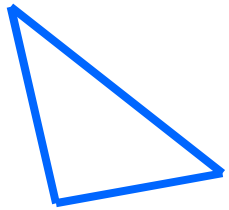
- **Az if else szerkezet** *kétirányú elágazást tesz lehetővé. Alakja:*

```
if ( <kifejezés> )
   <utasítás1>
else
   <utasítás2>
```

Ha a <kifejezés> igaz (nem nulla) akkor az <utasítás1>, egyébként az <utasítás2> hajtódik végre, majd a programfutás az <utasítás2> utáni programrészen folytatódik.

Elágazásszervező utasítások . .

Példa az if else szerkezet *alkalmazására*



```
/* Háromszög */
#include <stdio.h>
main()
{
    unsigned int a,b,c;
    printf("Háromszogtipus meghatarozasa\n\n");
    printf("Adja meg az oldalakat csokkeno sorrendben ! \n");
    printf("A= ");
    scanf("%u",&a);
    printf("B= ");
    scanf("%u",&b);
    printf("C= ");
    scanf("%u",&c);
```

- **Elágazásszervező utasítások . .**
  - **Példa az if else szerkezet** *alkalmazására ..*

```
if (a < b + c)
  {
    if (b * b + c * c == a * a)
    {
      if (b == c)
      {
        printf("egyenlőszárú derékszögű ");
      }
      else
      {
        printf("derékszögű ");
      }
    }
    else if (a == c && c == b)
    {
      printf("egyenlő oldalú (szabályos) ");
    }
    else if (a == b || b == c)
    {
      printf("egyenlőszárú ");
    }
    else
    {
      printf("általános ");
    }
  }
  else
  {
    printf("nem alkotnak háromszöget ");
  }
} /* Megj: egyenlő szárú derékszögű csak valós oldalhosszal teljesül.*/
```

Elágazásszervező utasítások . .

A switch többirányú elágaztató utasítás

Többirányú elágaztatás egymásbatokozott if else *szerkezetekkel is szervezhető, amint az előző példa mutatta. Azonban olyan esetekben, amikor egy egész jellegű kifejezés értékétől függően kell más-más programrészt végrehajtani, a* switch *utasítás áttekinthetőbb szerkezetet eredményez. Az utasítás alakja:*

```
switch ( <egész jellegű kifejezés> )
{
 case <konstans_kifejezés1> : <utasítások>
 case <konstans_kifejezés2> : <utasítások>
  ...
 default : <utasítások>
}
```

Elágazásszervező utasítások . .

A switch többirányú elágaztató utasítás működése

Amennyiben található az utasítás blokkjában a ***case*** *után álló <konstans_ kifejezés>-ek között az <egész jellegű kifejezés> értékével azonos érték, akkor a program végrehajtása az azt követő utasításokon folytatódik.*

Ha nincs ilyen értékű ***case*** *címke, akkor a programfutás a* ***default*** *címke utáni utasításokkal folytatódik.*

Amennyiben nincs ***default*** *címke, akkor a programfutás ilyen esetben a blokk utáni utasítással folytatódik. A* ***default*** *címke állhat középen, vagy elöl is.*

Azonos ***case*** *címkék nem adhatók meg.*

A ***switch*** *utasítás általában az* ***if else*** *szerkezettől gyorsabb futást eredményez.*

Elágazásszervező utasítások . .

Példa switch utasítással

```
#include <stdio.h>
main()
  {
    int i ;
    printf("Hány szót írjak ki? (max.3)=" );
    scanf( "%d", &i );
    switch ( i )
    {
        case 3: printf( " Alma");
        case 2: printf(" Körte");
        case 1: printf( " Barack");
    }
  }
```

Elágazásszervező utasítások . .

Break *utasítás a* switch *utasításban*

Sokszor hátrányos az, hogy nem csak a kifejezés értékének megfelelő **case** *címkéhez tartozó utasítások hajtódnak végre, hanem a blokk végéig elhelyezkedő összes többi is. Ezt elkerülhetjük a* **break** *utasítás alkalmazásával, amely a* **switch** *utasítás blokkját követő utasításra ugrat. Erre ad példát a következő program:*

```
# include <stdio.h>
int main() {
    char operator;
    float firstNumber, secondNumber;

    printf("Kerem az muveletet (+, -, *, /): ");
    scanf("%c", &operator);

    printf("Irjon be ket szamot: ");
    scanf("%f %f", &firstNumber, &secondNumber);
```

Elágazásszervező utasítások . .

Break *utasítás a* switch *utasításban ..*

```
switch(operator)
  {
    case '+':
      printf("%f + %f = %f", firstNumber, secondNumber, firstNumber + secondNumber);
      break;
    case '-':
      printf("%f - %f = %f",firstNumber, secondNumber, firstNumber - secondNumber);
      break;
    case '*':
      printf("%f * %f = %f",firstNumber, secondNumber, firstNumber * secondNumber);
      break;
    case '/':
      printf("%f / %f = %f",firstNumber, secondNumber, firstNumber/secondNumber);
      break;
    default:
      printf("Hiba! Nincs ilyen muvelet");
  }
}
```

Elágazásszervező utasítások . .

- **A goto utasítás** *alkalmazása struktúrált programokban ritkán fordul elő. Használata csak akkor célszerű, ha nélküle bonyolultabb programszerkezet adódna, pl. egymásba ágyazott ciklusokból való kiugrás a ciklusok lefutása előtt.*

  Alkalmazásához címkét kell elhelyezni a program azon pontján, ahová ugrani szeretnénk.

  A címkével ellátott utasítássor alakja:

  <azonosító> **:** <utasítás>

  és a

  goto <azonosító>;

  utasítással ugrathatunk feltétel nélkül az utasítássorra, amelynek a **goto***-val azonos függvényben kell lennie.*

Ciklusszervező utasítások

- *A ciklusszervező utasítások szolgálnak ugyanazon utasítások általában eltérő adatokkal történő többszöri végrehajtására. A C nyelvben két elöltesztelős (**while** és **for**) és egy hátultesztelős (**do while**) ciklusszervező utasítás használható, melyek futása még további feltétel nélküli vezérlésátadó utasításokkal (**goto**, **break**, **continue**) módosítható. Ezen utóbbi lehetőségek használata nem eredményez struktúrált programot.*

- **A while ciklusszervező utasítás**:

while (<kifejezés>)
<utasítás>

- *Minden egyes ciklusban előbb kiértékelésre kerül a <kifejezés> és csak akkor hajtódik végre a ciklus magját jelentő <utasítás>, ha a <kifejezés> igaz (nem nulla értékű). Ha a <kifejezés> hamis (értéke nulla), akkor a program futása a **while** utasítás utáni utasítással folytatódik.*

Ciklusszervező utasítások . .

- *A **while** használata akkor célszerű, ha a ciklusmag előre ismeretlen számban - lehet, hogy egyszer sem - hajtódhat végre.*

Pl.:

```
#include <conio.h>        /* getch miatt */
#include <stdio.h>        /* printf, scanf miatt */
main()
{
  float adat, osszeg = 0;
  while ( printf("Van adat? (I/N):\n") , getch() == 'i' )
  {
     printf("Adat=");
     scanf("%f", &adat );
     szumma += adat;
  }
  printf("Az osszeg= %f", osszeg);
}
```

Ciklusszervező utasítások . .

- **A for ciklusutasítás** *legelőnyösebben akkor használható, ha a végrehajtandó ciklusok száma előre ismert. A C nyelv* **for** *ciklusa sokféle alakot ölthet. Ez szerkezetéből is látható:*

```
for ( <kezdőért_kif>;<feltétel_kif>;<léptető_kif>)
    <utasítás>
```

Bármelyik kifejezés elmaradhat. Ha a <feltétel_ kif> *hiányzik, igaz értéket helyettesít a program.*

Működése: *legelőször egyszer kiértékelődik a* <kezdőért_kif>*, majd ciklusban kiértékelődik a* <feltétel_kif>*, végrehajtódik az* <utasítás> *és kiértékelődik a* <léptető_kif>*, mindaddig, amig a* <feltétel_kif> *igaz (értéke nem nulla). Ha a* <feltétel_kif> *nem igaz, a program futása a* **for** *ciklus utasítását követő program-utasításon folytatódik.*

Ciklusszervező utasítások . .

- Példa **for** ciklusutasításra

```
#include <stdio.h>
main()
{
    int szumma, k ;
    for (szumma = 0, k = 1 ; k <= 100 ; k++)
    {
        szumma += k;
    }
    printf("Szamok osszege szazig= %d", szumma);
}
```

1+2+3+4+5+6+7+8+9+10+11+12+ .. +92+93+94+95+96+97+98+99+100

Más megoldás:

```
for( k = 0; k<100; k++)
{
   szumma += k+1;
```

Ciklusszervező utasítások . .

- **A do while ciklus** *előnyös alkalmazási területe az előre nem ismert számban, de legalább egyszer végrehajtódó ciklustörzsű ciklusok. Alakja:*

```
do
 <utasítás>
while ( <kifejezés> ) ;
```

A ciklusban először végrehajtódik az <utasítás>*, majd kiértékelődik a* <kifejezés>*. Ha a* <kifejezés> *igaz (nem nulla), akkor az előbbiek ismétlődnek. Ha a* <kifejezés> *nem igaz (értéke nulla), a programvégrehajtás a* **do while** *utasítás utáni utasításon folytatódik.*

Ciklusszervező utasítások . .

- Példa **do while** ciklusra

```
#include <stdio.h>
#include <time.h>
main()
{
  int szam, tipp, n = 0;
  srand(time(NULL));
  szam = rand() % 10;
  szam++;
  printf("Gondoltam egy szamot 1-10 között \n %d", szam);
  do
   {
     printf("\nA tippje= ");
     scanf("%d",&tipp);
     n++;
   }
  while ( tipp != szam );
  printf("%d lepesben kitalalta!", n);
}
```

Ciklusleftutás módosítása: break, continue utasítások

- *A struktúrált programok írása során egyedül a* **break** *utasítást alkalmazhatjuk a* **switch** *utasításban. Egyes esetekben az egyszerűbb programszerkezet eléréséért feláldozhatjuk a struktúrált szerkezetet a* **break**, *ill. a* **continue** *alkalmazásával.*

  A **break** *utasítás feltételnélküli vezérlésátadást (kiugrást) valósít meg a* **break**-*et tartalmazó legbelső* **while**, **for** *vagy* **do while** *utasítás ciklusmagjából a ciklusutasítást követő programutasí tásra, ilymódon lehetővé téve a ciklus elhagyását annak teljes lefutása nélkül.*

  A ciklusmagban elhelyezkedő **continue** *utasítás a ciklusmag teljes végrehajtása helyett átugorja a ciklusmag* **continue** *utáni részét, majd folytatja a ciklusutasítás végrehajtását.*

Ciklusleftás módosítása: break, continue utasítások

```
#include <stdio.h>
#include <time.h>
main()
{
    int t;
    for ( ; ; )  // végtelen ciklus
    {
        scanf("%d", &t);
        if ( t == 10 )
        {
            break;  // itt lépünk ki
        }
    }
}
```

```
#include <stdio.h>
int main()
{
   for (int j=0; j<=10; j++)
   {
      if (j == 5)
      {
        continue; // innen elölről kezdjük
      }
      printf("%d ", j);
   }
}
```