Mérnöki programozás 5

Szerkesztette: dr. Vass Péter Tamás

Vezérlési szerkezetek

A vezérlési szerkezetek kiemelten fontos részei a C programozási nyelvnek. Alkalmazásukkal határozhatjuk meg a program utasításainak végrehajtási sorrendjét.
A vezérlési szerkezeteket felépítő szintaktikai elemek:

- kulcsszavak (pl. if, else, while stb.),
- utasításblokkot határoló kapcsos zárójelpár { },
- a vezérléshez kapcsolódó feltételt és egyéb utasítást tartalmazó zárójel pár ( ),
- utasításlezáró jel ;

A vezérlési szerkezetek alkalmazásának két fő célja:

- feltételes elágazások és
- ciklusok megvalósítása a programban.

Vezérlési szerkezetek

Feltételes elágazások megvalósítására szolgáló vezérlési szerkezetek

1. Utasítás, vagy utasítások feltételes végrehajtása

```
if (kifejezés) utasítás
```

vagy

```
if (kifejezés){
        utasítások
}
```

Csak a kifejezésben megfogalmazott feltétel teljesülése esetén kerül végrehajtásra az utasítás (ill. az utasításblokkba foglalt utasítások). Egyébként pedig a program végrehajtása átugorja az utasítást, ill. utasításblokkot, és a szerkezet utáni első utasítás kiértékelése következik.

Igaznak bizonyul a feltétel, ha a kifejezés értéke nem nulla. Csak akkor hamis a feltétel, ha a kifejezés értéke nulla.

Vezérlési szerkezetek

Példák:

```
if (a > b) x++;
```

```
if (a > b){
   x++;
   y = 2 * x;
}
```

2. Kétirányú elágazás megvalósítása

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

A kifejezésben megfogalmazott feltétel teljesülése esetén az ifághoz tartozó utasítás vagy utasításblokk kerül végrehajtásra.
Ellenkező esetben pedig az elseághoz tartozó utasítást vagy utasításblokkot hajtja végre a program.

Vezérlési szerkezetek

Példák:

```
if (a > b)
    x++;
else
    x--;
```

```
if (a > b){
    x++;
    y = 2 * x;
}
else{
    x--;
    y = 4 * x;
}
```

Vezérlési szerkezetek

3. Többirányú elágazás megvalósítása

```
if (kifejezés1)
        utasítás1
else if (kifejezés2)
        utasítás2
…..
else
        utasítás
```

Ha az ifághoz tartozó kifejezésben megfogalmazott feltétel igaz, akkor az if ág utasítása vagy utasításblokkja kerül végrehajtásra.
Ha nem teljesül az if ág feltétele, akkor a soron következő else if ág feltétele kerül kiértékelésre.
Egymás után több else if ág is beiktatható a szerkezetbe. Mindegyik saját feltétellel rendelkezik.
Ha valamelyik feltétel teljesül, akkor a hozzá tartozó ág utasítása, ill. utasításai végrehajtódnak, és a többi ág már nem kerül kiértékelésre.
Ha egyik feltétel sem teljesül, akkor az else ág tartalmát hajtja végre a program.

Vezérlési szerkezetek

Példa:

```
if (x < 0)
    x++;
else if (x > 0)
    x--;
else
    ;
```

Az önmagában álló utasításlezáró jel egy *üres utasítást* jelent. Ilyenkor semmilyen művelet nem kerül végrehajtásra az adott ágon.
Ha több utasítást tartalmaz valamelyik ág, akkor utasításblokkba kell foglalni { }.

Vezérlési szerkezetek

4. Többirányú elágazás switch – case szerkezettel

```
switch (kifejezés){
        case konstans kifejezés1:
                utasítás(ok)1
        case konstans kifejezés2:
                utasítás(ok)2
        …..
        default:
                utasítás(ok)
}
```

A switch – case vezérlési szerkezet olyankor használható, amikor a switch kulcsszó utáni kifejezés értékétől függően akarunk más – más utasításokat végrehajtatni a programmal. Az egyes case címkék utáni konstans kifejezések értékeivel hasonlítja össze a program a switch kifejezésének értékét. Ha egyezést talál, akkor az adott caseághoz tartozó utasítások kerülnek végrehajtásra. Ha egyik case címkéhez tartozó konstans kifejezés értéke sem bizonyul egyenlőnek a switch kifejezés értékével, akkor a default címkével azonosított ág utasítása, ill. utasításai kerülnek végrehajtásra.

Vezérlési szerkezetek

```c
#include <stdio.h>
main(){
    int n;
    printf("Adjon meg egy 10-nel kisebb pozitiv szamot: ");
    scanf("%d", &n);
    if (n>10 || n<1)
        printf("Helytelen adat!\n");
    else{
        switch (n){
            case 1:
                printf("A megadott szam 1.\n");
                break;
            case 2:
                printf("A megadott szam 2.\n");
                break;
            case 3:
                printf("A megadott szam 3.\n");
                break;
            default:
                printf("A megadott szam nagyobb, mint 3.\n");
                break;
    }
    printf("A program vege.\n");
}
```

Vezérlési szerkezetek

4. Többirányú elágazás switch – case szerkezettel

Az egyes case címkék utasítása, ill. utasításai után alkalmazott break utasítások azt eredményezik, hogy amennyiben egy adott konstans kifejezés értéke egyenlőnek bizonyul a switch kifejezésének értékével, akkor a megfelelő case címke ágához tartozó utasítás(ok) végrehajtása után a többi case címke konstans kifejezéseire vonatkozóan már ne végezzen vizsgálatokat a program.

A break utasítása nélkül, minden egyes case címkéhez rendelt konstans kifejezés értékét összehasonlítja a program a switch kifejezés értékével még akkor is, ha már talált egyezést az egyik ágon.

Természetesen csak annak az ágnak az utasításai kerülnek végrehajtásra, amire a kifejezések egyenlősége teljesül.

A case címkékhez tartozó utasításokat nem kell blokkba foglalni kapcsoszárójelekkel, de a kettőspont szimbólum alkalmazása kötelező a címke kifejezése után.

Vezérlési szerkezetek

5. Elöltesztelő while ciklus

```
while (kifejezés)
        utasítás
```

vagy

```
while (kifejezés){
        utasítások
}
```

Az előzőekben már megismertük!

6. Hátultesztelő do - while ciklus

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

vagy

```
do {
        utasítások
} while (kifejezés);
```

Az előzőekben már megismertük! Ne felejtsük el kitenni az utasításlezáró jelet (pontosvessző) a while kifejezése után!

Vezérlési szerkezetek

7. for ciklus

```
for (inicializáló kifejezés; feltétel; léptető kifejezés)
    utasítás
```

vagy

```
for (inicializáló kifejezés; feltétel; léptető kifejezés){
    utasítások
}
```

A for után következő *inicializáló kifejezés*ben kezdő értéket rendelünk hozzá a *ciklusváltozó*hoz (pl. i = 0).
A következő kifejezés a ciklusváltozó lehetséges felső vagy alsó értékhatárára vonatkozó feltételt tartalmaz (pl. i <= 8).
A *léptető kifejezéssel* növelhetjük vagy csökkenthetjük a ciklusváltozó értékét a ciklus magjának minden egyes végrehajtása után (pl. i=i+2).

Minden for ciklus megvalósítható while ciklussal, de fordítva ez nem igaz!

Vezérlési szerkezetek

break utasítás

A break utasítást nem csak a switch – case szerkezetnél lehet alkalmazni, hanem a ciklusok esetében is.

Ha a break utasítást valamilyen feltétel teljesüléséhez kötve alkalmazzuk egy ciklusmagon belül, akkor lehetővé válik a kiugrás a ciklusból mielőtt még a ciklusfeltételnek megfelelő számú ismétlődés végbemenne.

continue utasítás

A continue utasítás szintén alkalmazható a ciklusmagon belül. Ezt is valamilyen feltétel teljesüléséhez kötve használjuk leginkább (if, if-else stb.). Hatása annyiban tér el a break utasításétól, hogy a continue utasítás nem eredményezi a ciklus idő előtti leállását. Ha a hozzárendelt feltétel teljesül, akkor az utána következő utasítások a ciklusmagon belül nem hajtódnak végre az adott ismétlődési lépésben. A ciklus végrehajtása a ciklusfeltétel kiértékelésére ugrik azonnal.

Vezérlési szerkezetek

return utasítás

A return utasítás befejezi az éppen végrehajtás alatt álló függvény futását. Ezután a program vezérlése (utasításvégrehajtási folyamat) visszakerül ahhoz a függvényhez (*hívó függvény*), amelyikben a leállított függvény (*hívott függvény*) meghívásra került. Ezt az utasítást is gyakran alkalmazzák feltételes szerkezethez kötve (if, if-else stb.)
Ha a main függvényben (főprogram) alkalmazzuk a return utasítást, akkor a program futása leáll a végrehajtásakor.
A return utasításhoz kifejezés is rendelhető, melynek értéke lesz a leállított függvény un. *visszatérési értéke*. Ezt az értéket a hívó függvény kapja meg, és felhasználhatja a futása során.
Természetesen a return után következő kifejezés megadásakor ügyelni kell arra, hogy a lehetséges értékei összhangban legyenek a függvény definíciójában előírt kimeneti paraméter típusával (ld. függvények definíciója).

Vezérlési szerkezetek

```c
#include <stdio.h>
/*A program két szamot var a bemeneten es szamolja a
hanyadosukat.*/
main(){
   float a, b;
   char c;

   printf("Adja meg az osztando erteket: ");
   scanf("%f",&a);
   printf("Adja meg az oszto erteket: ");
   scanf("%f",&b);
   if (b==0.0){
      printf("A 0-val torteno osztasnak nincs ertelme!");
      return;
   }
   printf("Az osztas eredmenye: %f\n", a/b);
}
```

Vezérlési szerkezetek

```
#include <stdio.h>

main(){
/*A program beolvas egy sort, és megszámolja a
karaktereket*/
	int nc=0;
	char c;

	printf("Gepeljen be egy sort majd nyomjon \
enter-t!\n");
	do{
		c = getchar();
		nc++;
		putchar(c);
	}while ( c != '\n');
	printf("A karakterek szama: %d\n", --nc);
}
```

Vezérlési szerkezetek

```
#include <stdio.h>
#include <ctype.h>
/*A program beolvas egy sort, és megszámolja az első
üres karakter előtti karaktereket számát*/
main(){
	int nc=0;
	char c;

	printf("Gepeljen be egy sort majd nyomjon \
enter-t!\n");
	do{
		c = getchar();
		if (isspace(c))
			break;
		nc++;
		putchar(c);
	}while ( c != '\n');
	printf("\nAz elso ures karakter elotti \
karakterek szama: %d\n", nc);
}
```

Vezérlési szerkezetek

```c
#include <stdio.h>
#include <ctype.h>
/*A program beolvas egy sort, és megszámolja a nem üres
karakterek számát a sorban*/
main(){
	int nc=0;
	char c;

	printf("Gepeljen be egy sort majd nyomjon "\
		"enter-t!\n");
	do{
		c = getchar();
		if (isspace(c))
			continue;
		nc++;
		putchar(c);
	}while ( c != '\n');
	printf("\nA nem ures karakterek "\
		"szama: %d\n", nc);
}
```

Vezérlési szerkezetek

Magyarázat

A getchar( ) és a putchar( ) függvények a szabványos bemeneti és kiviteli függvénykönyvtár részei (hasonlóan a printf és a scanf függvényekhez). Hívásukhoz az stdio.h fejlécfájlt be kell építeni a forráskódba az előfordítónak szóló include kulcsszóval.

```
c = getchar();
```

A getchar függvény beolvas egy karaktert a parancssorból, és a karakter kódjával tér vissza. A karakter kódját a fenti utasítás szerint a c változóban tároljuk el, amit előzőleg char típussal deklaráltunk.

```
putchar(c);
```

A putchar függvénynek egyetlen bemeneti paramétere van, amelyben egy karakter kódját képes fogadni. A feladata, hogy a megadott kódú karaktert megjelenítse a parancssorban.

Vezérlési szerkezetek

Magyarázat

`isspace(c)`

Az isspace is egy szabványos könyvtári függvény, amelynek deklarációját a ctype.h fejlécfájl tartalmazza. A hívásához tehát a fejlécfájlt be kell építeni a forráskódba az előfordítónak szóló include kulcsszóval.

Ez a fejlécfájl tartalmazza a karakteres vizsgálatokra szolgáló összes függvények deklarációját.
A karakteres vizsgálatokra használható könyvtári függvényeknek egyetlen karakter típusú bemeneti paramétere van. Ebben fogadják annak a karakternek a kódját, amelyre nézve valamilyen feltétel teljesülését vizsgálják.
Ha a feltétel nem teljesül, akkor 0, azaz logikai hamis értékkel térnek vissza.
Ha a feltétel teljesül, akkor nem 0, azaz logikai szempontból igaznak minősülő értéket adnak a kimenetükön.

Vezérlési szerkezetek

Magyarázat

`isspace(c)`

Hívásakor az isspace függvény a c változóban tárolt karakter kódját kapja meg a bemenetén, és megvizsgálja, hogy a karakter megegyezik-e valamelyik üres karakterrel (szóköz, horizontális vagy vertikális tabulátor, újsor, lapdobás).

Ha egyezést talál, akkor nem 0, azaz igaz logikai értékkel tér vissza a hívó függvényhez (ami a példaprogramban maga a main függvény).

Ha nem talál egyezést, azaz a vizsgált karakter nem üres karakter, akkor 0 (logikai hamis) lesz a visszatérési értéke.

A példákban a függvényhívás egy if szerkezet feltételében helyezkedik el. Ez azt eredményezi, hogy az if –hez tartozó ág utasításai csak akkor kerülnek végrehajtásra, ha a vizsgált karakter üres karakternek minősül.

Vezérlési szerkezetek

Néhány további karakterviszgáló könyvtári függvény

islower(c)	értéke igaz, ha c egy kisbetű
isupper(c)	értéke igaz, ha c egy nagybetű
isalpha(c)	értéke igaz, ha c egy betű
isdigit(c)	értéke igaz, ha c egy decimális számjegy
isprint(c)	értéke igaz, ha c egy nyomtatható karakter, beleértve a szóközt is
ispunct(c)	értéke igaz, ha c egy nyomtatható karakter, de nem betű, számjegy, vagy szóköz
iscntrl(c)	értéke igaz, ha c egy vezérlőkarakter

A ctype.h-ban deklarált karakter átalakító függvények

int tolower(c)	a betűt képviselő c karaktert kisbetűvé alakítja, és a kisbetű karakterkódjával tér vissza
int toupper(c)	a betűt képviselő c karaktert nagybetűvé alakítja, és a nagybetű karakterkódjával tér vissza