C programozás

6 óra

Függvények, függvényszerű makrók, globális és lokális változók

1.Azonosítók

A program bizonyos összetevőire névvel (azonosító) hivatkozunk

Első karakter: _ vagy betű (csak ez lehet, kis és nagy betű számít !!!)

pl.: szam, _alap

2. Deklaráció és definíció

a.) **Minden névről meg kell mondani mire szeretnénk használni, mert így a fordító nem tud mit kezdeni az adott névvel. → DEKLARÁCIÓ**

A név tulajdonságait (típus, tárolási osztály, láthatóság) közöljük a fordítóval.

b.) **Ha az a cél, hogy a deklarációnak megfelelő objektum is létrejöjjön a memóriában akkor DEFINÍCIÓT kell alkalmazni.**

Tehát a DEFINÍCIÓ olyan DEKLARÁCIÓ, amely helyfoglalással jár.

Objektum: **olyan memóriaterület jelöl, amely egy vagy több értéket tartalmaz.**

3. Függvények

Mire jók a függvények”

- tagolásra, struktúráltság növelésére

- külső, könyvtári függvények használatára

Definíció <u>nem</u> Deklaráció

Függvénydefiníció alakja:

<visszatérési típus> név (<paraméterlista>)

{ <lokális definíciók és deklarációk>

<utasítások>

}

Pl.:

```
double  LegkisebbTombben  (double t[],  int n)
{  int x;
   double min;

   for (min = t[0], x=1;   x < n;   x++)
      if (t[x] < min)   min = t[x];
   return min;                  /* visszatérés és érték megadása */
} /* LegkisebbTombben () */
```

Hívása pl.:

```
main ()                     /* ez is egy speciális függvény ... */
{       ...
   #define tombmeret 2000
   double dt [tombmeret], mm;
   int x;
   ...
   x= ....;
   mm = LegkisebbTombben (dt,x*2-1);   /* fv. hívása */
   ...
}
```

Függvénydeklaráció alakja:

<visszatérési típus> név (<paraméterlista>);

azaz csak a függvényfejet adjuk meg, utána pontosvessző.

Ezt nevezik fv. prototípusnak is. Pl.:

```
double Egyik (int p1,  double p2);
double Egyik (int,  double);
```

megadja azokat az információkat, melyek a függvény hívásához kellenek: függvény neve, visszatérési típusa, paraméterek száma és típusa, de nem adja meg a függvény által végrehajtandó utasításokat.

```
double Egyik (int p1, double p2);
```
; van a végén: deklaráció

```
void Masik (char *p1, unsigned t[])
{ ....
  if (Egyik( .....) >0) ....
                        ....
} /* Masik() */
```
: definíció, mert blokk követi:

: **Egyik** hívása

```
double Egyik (int p1,  double p2)
{  ....
   Masik ( ..... );
   ....
} /* Egyik() */
```
: **Egyik** definíciója

: **Masik** hívása

C-ben csak értékparaméter van!

Visszaadott érték:

- függvény visszatérési értékével
- globális, azaz file-szintű változóval
- pointer formális-, cím aktuális argumentum segítségével:

Pl. tömbben legkisebb és legnagyobb elem meghatározása:

nincs visszatérési értéke:

```
void  MinMax (double t[], int n, double *minp, double *maxp)
{  int x;

   *minp = *maxp = t[0];   /* első elem értéke a minp és maxp
                              által mutatott helyekre kerül */
   for (x=1;  x<n;  x++)  {
      double tx=t[x];          /* blokk lokális változója */
      if (tx < *minp) *minp=tx;  else
      if (tx > *maxp) *maxp=tx;
   } /* for x */
   return;                     /* itt elhagyható */
}
```

Hívása pl.:

```
ff (....)                          :egy másik függvény, MinMax után
{
   double dd [....], min, max;
   ...
   MinMax (dd, 100, &min, &max);   :dd, min és max címét adja át,
                          az eredmény min-be és max-ba kerül
   ...
} /* ff() */
```

`MinMax()` végrehajtása alatt:

minp ***minp ≡ min** **maxp** ***maxp ≡ max**

Megjegyzés: a pointer-tömb rokonság miatt a függvény feje lehetne ilyen is:

```
void  MinMax (double *t, int n, double *minp, double *maxp)
```

Függvények további szabályai

A visszatérési érték alapértelmezése int, lehet pointer, de nem lehet függvény vagy tömb.

Pl.:

```
ff (float f) ....   ≡  int ff (float f) ....
```

Függvény csak file-szinten definiálható, blokkban lokális függvény nem definiálható, de deklarálkató, pl.:

```
int f2 (double p) {                          : file szintű definíció, benne:

   void f3 (float fp);                       : lokális függvény Deklaráció: OK.

   int Negyzet (int x)
      {  return x*x; }                       : HIBA: fv. definíció csak file szinten lehet
   ....
   f3 (3.2*f1("szöveg"));                    : f3 és f1 hívása
}

char *f4 (void)  {                           : paraméter nélküli függvény
   ....
   f3(4.1F);                                 : HIBA: f3 itt nem ismert
   ....
}

void f3 (float fp) { ....}                   : f3 definíciója
```

Egymást követő, azonos típusú formális paraméterek típusát külön-külön ki kell írni, pl.:

```
void fx (int p1, p2)          HIBÁS!!!
void fx (int p1, int p2)      Helyes
```

Paraméter nélküli függvény definíciója illetve deklarációja pl.:

```
void f4 (void) ....
```
: paraméter nélküli függvény

Ennek hívása:

```
....
f4()
....
```
: ki kell írni: **()**

Deklarációban a paraméternevek elhagyhatóak. pl.:

```
double Egyik (int, double);
```
: paraméter név itt nem fontos,
de definícióban igen

Deklaráció alakja csak prototípus lehet:

```
void f5 (int p1, char p2);
```

Rossz: `void f5 (p1,p2);  int p1; char p2;`

3.Függvényszerű makrók

```
#include <stdio.h>
#define sqr(X) ((X) * (X))
#define max(X,Y) ((X) > (Y) ? (X) : (Y))
#define abs(X) ((X) < 0 ? (-(X)) : (X))
```

```
#define sqr(X) ((X) * (X))
#define sqr1(X) (X * X)
```

```
                  a                              a+1
sqr       ( (a) * (a) )                ( (a+1) * (a+1) )
sqr1        ( a * a )                    ( a+1 * a+1 )
```