A C programozási nyelv

- *A C nyelv kialakulása*
- *Egy egyszerű C program*
- *A C nyelv jellegzetességei*
- *A C program összetevőinek felépítése*
  - ***Deklarációk***
  - ***Blokk***
  - ***Függvénydefiníció***
- *A C program felépítése*
- *Típusok, változók, konstansok*
  - ***Változók definiálásának egyszerű esetei***
  - ***Saját típus definiálása***
  - ***Konstansok megadása***
- *Be- és kiviteli függvények egyszerű alakja*
  - ***Adatbeolvasás formátumellenőrzéssel***
  - ***Egyetlen karakter beolvasása megjelenítés nélkül***
- *Cím, érték, mutató fogalma*

A C nyelv kialakulása

Nevezetes mérföldkövek

BCPL nyelv, *Martin Richards*, 1963
B nyelv, *Ken Thompson*, 1970 ⇒ UNIX megírására
C nyelv, *Dennis M. Ritchie*, 1971 ⇒ hatékonyabb verzió
"The C programming language" referenciamű,
B.W.Kernighan és D.M.Ritchie
ANSI C szabvány, 1989, Amerikai Szabványügyi Hivatal

- **A „Szia világ” – első c nyelvű mintapélda**

```
#include <stdio.h>

int main()
{
    printf("Hello, world");
    return 0;
}
```

Egy lehetséges assembly változata

```
            JMP start
data:       DB "Hello, world"   ; Változó
            DB 0                ; Sztring lezáró nulla

start:
            MOV C, data         ; memóriacímre mutat
            MOV D, 0xE8         ; a kimenetre mutat
            CALL print
            HLT                 ; halt – kilépés a programból
print:
            PUSH A
            PUSH B
            MOV B, 0
loop:
            MOV A, [C] ; Betölti a következő karaktert
            MOV [D], A ; Kiírja a betöltött karaktert
            INC C      ; C = C + 1
            INC D      ; D = D +1
            CMP B, [C] ; 0-t olvastunk be?
            JNZ loop   ; ugorj vissza ha nem nulla

            POP B
            POP A
            RET
```

https://schweigi.github.io/assembler-simulator/

Egy „bonyolultabb” C program

```
#include <stdio.h>        /* megjegyzes: printf, scanf miatt */

#define  PI  3.1415926

int main()
{
  float r, kerulet, terulet;
  printf( "Kerem a kor sugarat [meter] = " );
  scanf( "%f" , &r );      // specialis beviteli fuggveny
  kerulet = 2 *r * PI;
  terulet = r * r * PI;
  printf( "A kerulet= %f meter \n", kerulet );
  printf( "A terulet= %f negyzetmeter\n", terulet );
  return 0;
}
```

A C nyelv jellegzetességei

- *Általános célú, magasszintű nyelv*
- *Tömör, gyors programok írására alkalmas*
- *Gépfüggetlensége, hordozhatósága jó*
- *Különféle hardverplatformokon létezik*
- *Rendszerprogramozásra alkalmas*
- *Az eljárásokat típus nélküli (void) függvényekkel valósítja meg*
- *Alprogram nem definiálható alprogramon belül, csak deklarálható*
- *Erősen épít a mutatók használatára, speciális pointeraritmetikát alkalmaz*

- **A C nyelv jellegzetességei . .**
  - *Lehetőséget nyújt makro-k (automatikus szöveghelyettesítés) használatára*
  - *Sztring típus helyett karaktervektorokat alkalmaz*
  - *Gazdag függvénykönyvtár készlettel rendelkezik*
  - *PC-ken többféle memóriamodell szerinti fordítás közül választható a leggazdaságosabb*
  - *Lehetővé teszi egyes változóknak a processzor regisztereihez rendelését*
  - *A fordításkori optimalizálás gazdag lehetőségét nyújtja*
  - *A kisbetűket és nagybetűket megkülönbözteti.*

A C program összetevőinek felépítése

- *Deklarációk* *(bármely sor elmaradhat, vagy többször ismétlődhet)*

```
#include <headerfájl-név.h>
#define <konstans, vagy függvénymakró>
typedef <típus> <újtípusnév>;          /* új típus definiálása  */
<típus> <változóazonosító>;            /* változódefiniálás */
<típus> <függvénynév(paraméterek)>; /* fv deklaráció */
```

- *Blokk*

```
{
  <deklarációk>    /* lokálisak, blokkon belül érvényesek  */
  <utasítások>     /* bármely utasítás lehet blokk */
}
```

A C program összetevőinek felépítése . .

- *Függvénydefiníció*

  <típus> <függvénynév(paraméterek)>
  <blokk>

- *A C program felépítése* *Megjegyzés: más felépítés is lehet*

```
<deklarációk>          /*globálisak, programon belül érvényesek */
main()                 /* a fő függvény  */
<blokk>                /* a fő függvény blokkja */
<függvénydefiníciók>   /* a deklarált fv-ek teljes megadása*/
```

Típusok, változók, konstansok

Típusnév	Értéktartomány	Formátumsztring (scanf és printf)	Méret
char	'\0',..,'!',..,'0','1',..,'A',..,'a',..,'\xFF'	"%c"	1- byte
signed char	-128..127	"%d"	1- byte
unsigned char	0..255	"%u"	1- byte
int	-32768..32767	"%d"	2 - byte
int	-2147483648..2147483647	"%d„	4 - byte
unsigned int	0..65535 /	"%u"	2/4 - byte
short int	-32768..32767	"%d"	2- byte
unsigned short int	0..65535	"%u"	2- byte
long	-2147483648..2147483647	"%ld"	4- byte
unsigned long	0..4294967295	"%lu"	4- byte
enum	felsorolás típus		
float	±(3.4E-38..3.8E38)	"%f" v. "%e„	4- byte
double	±(1.7E-308..1.7E308)	"%lf" v. "%le"	8- byte
long double	±(3.4E-4932..1.1E4932)	"%Lf" v. "%Le"	16- byte
mutató			
tömb			
struct			
union			

Változók definiálásának egyszerű esetei

<típus> <változó1>, <változó2>,...;
<típus> <vált1> = <kezdőért1>, <vált2> = <kezdőért2>,...;

Pl.: **int** *i, j, k* ;
float *valos* = 4.27, *beta* = 90, *delta* = 1E-03;

Saját típus definiálása

typedef <típus> <új_típus_neve>;

Pl.: **typedef** **unsigned long int** *rekordszamtip* ;

Konstansok megadása

Karakteres konstansértékek

- *általános karakterek:* pl.: 'c'

- *speciális karakterek:*

pl.:

'\0'	*sztringvégjel*
'\t'	*tabulátor*
'\n'	*új sorba lépés*
'\"'	*aposztróf* (')
'\\'	\ *jel*
'\r'	*sorelejére lépés*
'\xhh'	*hh hexadecimális kódú karakter*
'\oXX'	xx *nyolcas számrendszerbeli karakter*

Konstansértékű azonosítók

- *Változóból a const típusminősítő eléírásával*

  const <típuselőírás> <változónév> = <kezdőérték>;

  Pl.: **const float** *eps* = 0.003;

- *A #define előfordító direktívával*

  #**define** <azonosító> <helyettesítő szöveg>

  Pl.: #**define** PI 3.14159265
  #**define** TRUE 1
  #**define** FALSE 0
  #**define** SZOVEG "*Folytatás ?*"

Be- és kiviteli függvények egyszerű alakja

Adatbeolvasás formátumellenőrzéssel

scanf(<formátumsztring>, &vált1, &vált2,...);

A függvény a beolvasott adatot a formátumsztringben megadott típusú adatként értelmezi és konvertálja a változó típusára, majd elhelyezi a változó címére. A függvény visszatérési értéke a sikeresen beolvasott értékek száma (int), amelyet nem kötelező felhasználni. Az értékek elhelyezéséhez a változók címét kell megadni a & *operátorral.*

Pl.: **scanf**("%d" , &*egeszvaltozo*);
scanf("%f" , &*valosvaltozo*);
scanf("%u" , &*elojelnelkuli_egesz*);

Be- és kiviteli függvények egyszerű alakja . .

- *Egyetlen karakter beolvasása megjelenítés nélkül*

  karakterváltozó = **getch**();

  Pl.: **char** *bill* ;
  bill = getch();

- *Adatkiírás formázott módon*

  printf(<formátumsztring>, kifejezés1, kifejezés2,...);

  A kiíratás formátumsztringje tájékoztató szövegből és % jellel kezdődő konverziós előírásból állhat.

A konverziós előírás felépítése

Jelző: **–** *balraigazítás*

\+ *+ előjel kiíródik*

␣ *+ előjel helyett szóköz*

Méretmódosító betű:

h	**short int**,	*ha a konverziós betű: d, u*
l	**long int**,	*ha a konverziós betű: d, u*
l	**double**,	*ha a konverziós betű: f, e, E*
L	**long double**,	*ha a konverziós betű: f, e, E*

A konverziós előírás felépítése . .

Konverziós betű:

c	*a számnak megfelelő ASCII kódú karakter*
d	*előjeles egész*
u	*előjel nélküli egész*
f	**float**, *vagy* **double** *számot lebegőpontosként,*
e, E	**float**, *vagy* **double** *számot exponenciális alakban ír ki*
p	*mutató értékét adja szegmens:offset alakban*
s	*sztring karaktereit írja ki.*

Példák a konverziós előírás használatára

Pl.:

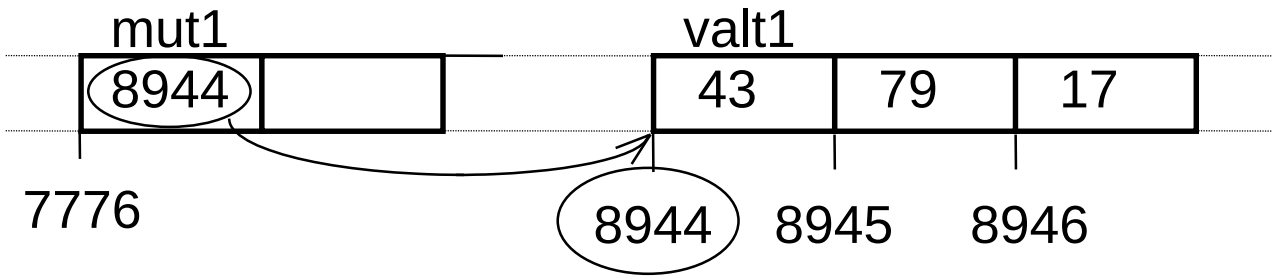
```
printf( "betü: %c " , 65 );          ⇒  betü: A
printf( "egész= %7d\n" , -125 );     ⇒  egész=     -125
printf( "Eredmény= %8.2f" , 62.872 ); ⇒  Eredmény=     62.87
printf( "exp.alak= %+8.2e" , 62.87 ); ⇒  exp.alak= +6.29e+01
printf( "Totál= %12lu" , 123456789 ); ⇒  Totál=     123456789
```

Cím, érték, mutató fogalma

*A memória egymást követő tárolóhelyek sorozata. Egy tárolóhelynek van sorszáma, amit **cím**nek nevezünk és van tartalma, amit az adott címen tárolt **érték**nek nevezünk. Mivel a címek számértékek és nehezen megjegyezhetők, ezért helyettesítjük azokat nevekkel, ezek a változók. Egy változó címe alatt az általa helyettesített címértéket értjük, értéke alatt pedig az adott címen tárolt értéket.*

*Amennyiben a cím tárolására akarunk változót, azaz egy másik címen található tárolóhelyet létrehozni, ezzel tulajdonképpen egy **mutató**t (pointert) hozunk létre, melynek számunkra az értéke fontos. A pointerváltozóban tárolt érték az a cím, amelyen az eredeti változónk található a memóriában.*

Cím, érték, mutató fogalma . .



A mut1 *pointerváltozónak értékül adhatjuk a* valt1 *változó címét a* & *operátor segítségével:*

mut1 = &*valt1* ;

ami után vált1 *tartalmát, értékét olvashatjuk, írhatjuk közvetett módon a* * (indirekció) *operátorral, mivel:*

valt1 == **mut1* .

Azaz valt1 *definiálása nélkül is tudjuk* mut1 *segítségével írni, olvasni a 8944-es című tárolóhely tartalmát, sőt egyszerűen* mut1 *tartalmának megváltoztatásával hozzáférhetünk pl. a 8945, vagy a 8946 tárolóhelyek tartalmához is.*