A C++ programozási nyelv 1.

Feladatmentes

2019.02.10

Koschek Vilmos

Miről lesz szó? Alapok…

1. Bevezetés
2. Típusok, operátorok és kifejezések
3. Vezérlési szerkezetek
4. Függvények és a programstruktúrák
5. Mutatók és tömbök
6. Összetett adatstruktúrák

1. rész

Bevezetés

Ajánlott irodalom

1. **Brian W.Kernighan – Dennis M. Ritchie: A C programozási nyelv**
2. **C.L.Tondo – S.E.Gimpel: C programozási gyakorlatok**
3. **Pethő Ádám: abC programozási nyelvkönyv**
4. **B.Stroustrup,The C++ Programming Language**
5. **S.B.Lippman,C++**

C nyelv

- Eredet
  - 1967 Martin Richards: BCPL, egyféle adattípus
  - 1970 Ken Thompson: B, PDP-7-n, UNIX alatt
  - 1972 Dennis Ritchie: Első C fordító, PDP-11
  - 1975 –től szinte minden gépen
  - CRAY-2, IBM System/370, Honeywell 6000, VAX, PS62, Next, IBM pc,…
- Általános célú programozási nyelv
  - Sokoldalú, tömör, viszonylag alacsony szintű
  - Megfelel a legtöbb rendszerprogramozási feladatra
  - Szinte mindenütt, és mindenen fut
  - Illeszkedik a UNIX programozási környezetre
  - dBase, MSC, Clipper, OS/2, UNIX (95%), CorelDraw,….
- Hordozhatóság

C++ nyelv

Eredet

1983:Bjarne Stroustrup C++ - C bővítése osztályokkal (1985-1.2, 1988-2.0)
1998:ISO/IEC 14882:1998, C++ 98 – Szabvány (egységesítés, könyvtár)
2003:ISO/IEC 14882:2003, C++ 03 - Javítások
2011:ISO/IEC 14882:2011, C++ 11 (újdonságok) ->C++14 -> C++17

C KOMPATIBILIS

- Jelentősen továbbfejlesztett és kiegészített C
- C előnyei (tömör, gépközeli, gyors, sokoldalú, szinte mindenen fut)
- Veszélyes konstrukciók biztonságossá tétele

OBJEKTUMORIENTÁLT

- Egységbezárás (encapsulation) adat+függvény -> objektum,osztályok
- Öröklés (inheritance) új osztálynál a régi felhasználása
- Többrétűség (polymorphism) virtuális függvények, operátorok és függvénynevek átdefiniálása

C++ nyelv

http://prog.hu/hirek/4049/18-idezet-a-programozasrol-hires-szakemberektol

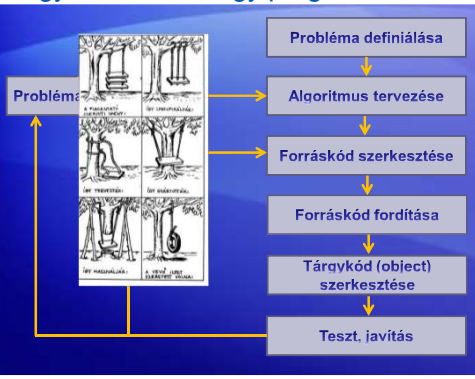
„…C++ nem a legkisebb és legtisztább nyelv, amelyet valaha terveztek. Mindazonáltal a C++

- elég tiszta ahhoz, hogy az alapfogalmakat sikeresen tanítsuk,
- elég valószerű, hatékony és rugalmas az igényes projektekhez is,
- elérhető olyan szervezetek és együttműködő csoportok számára, melyek eltérő fejlesztési és végrehajtási környezetekre támaszkodnak,
- elég érthető ahhoz, hogy bonyolult fogalmak és módszerek tanításának hordozója legyen és
- elég „kereskedelmi", hogy segítse a tanultak gyakorlatban való felhasználását.

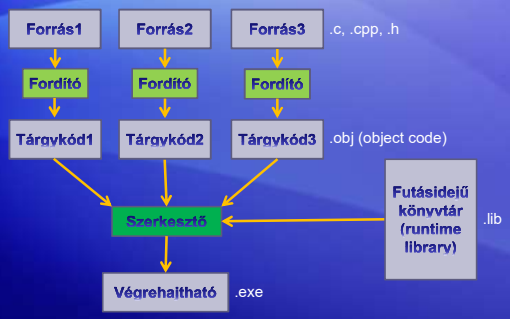
A C++ olyan nyelv, mellyel gyarapodhatunk. „

Bjarne Stroustrup, A C++ programozási nyelv

Hogyan is készül egy program?



És akkor valójában?



Néhány kiegészítés

- Előny
- Fordító (compiler) – kód
  - Előfeldolgozó, szintaktikai elemző
  - Kódgenerálás
  - Optimalizálás
- Szerkesztő (link) – hivatkozások
- Könyvtár (lib) – újrafelhasználás
- Lehetőségek
  - Compiler
  - Interpreter
  - Just-In-Time compiler

Egy példa

.C

```
int main(void)
{
        int i=10, j=20, k;

        k = f1(i,j);

        k = f2(i,j);
        return(0);
}
```

s1.c

```
int f1(int x, int y)
{
        int k;
        k = x + y;
        return(k);
}

int f2(int x, int y)
{
        int k;
        k = x - y;
        return(k);
}
```


OBJ

.OBJ (.c -> fordító -> .obj)

```
; 8    : k = f1(i,j);

0002c 8b45          mov eax, DWORD PTR _j$[ebp]
0002f 50            push eax
00030 8b 4d f8      mov ecx, DWORD PTR _i$[ebp]
00033 51            push ecx
00034 e8 00 00 00 00 call _f1
00039 83 c4 08      add esp, 8
0003c 89 45 e0      mov DWORD PTR _k$[ebp], eax
```

MAP

```
.MAP (szerkesztő)
 Start       Length     Name                   Class
 0001:00000000 00010000H .textbss               DATA
 0002:00000000 00003483H .text                  CODE
 0003:00000000 00000104H .CRT$XCA               DATA
 0003:00000104 00000104H .CRT$XCAA              DATA
 0003:00000208 00000104H .CRT$XCZ               DATA
 0003:0000030c 00000104H .CRT$XIA               DATA
 .
 .
 Address         Publics by Value    Base      Lib:Object
 .
 .
 0002:00000390   _main               00411390 f  main.obj
 0002:00000420   _f1                 00411420 f  s1.obj
 0002:00000460   _f2                 00411460 f  s1.obj
 .
```


Mit is jelent az optimalizálás?

main.c

```
#include <stdio.h>

int f1(int x, int y);
int f2(int x, int y);

int main(void)
{
    int i=10, j=20, k;

    k = f1(i,j);
    printf("\n%d", k);

    k = f2(i,j);
    printf("\n%d", k);

    return(0);
}
```

s1.c

```
int f1(int x, int y)
{
    int k;
    k = x + y;
    return(k);
}

int f2(int x, int y)
{
    int k;
    k = x - y;
    return(k);
}
```

Akkor tessék optimalizálni!

Optimalizálás nélkül

main.c

```
int i=10, j=20, k;
  mov DWORD PTR _i$[ebp], 10
  mov DWORD PTR _j$[ebp], 20

k = f1(i,j);
  mov eax, DWORD PTR _j$[ebp]
  push eax
  mov ecx, DWORD PTR _i$[ebp]
  push ecx
  call _f1
  add esp, 8
  mov DWORD PTR _k$[ebp], eax

printf("\n%d", k);
  mov eax, DWORD PTR _k$[ebp]
  push eax
  push OFFSET ??_C@_03IOBBOKCP@?6?$CFd?$AA@
  call DWORD PTR __imp__printf
  add esp, 8
```

s1.c

```
f1
int k;
k = x + y;
  mov eax, DWORD PTR _x$[ebp]
  add eax, DWORD PTR _y$[ebp]
  mov DWORD PTR _k$[ebp], eax

return(k);
  mov eax, DWORD PTR _k$[ebp]
```


Optimalizálva

2. rész

Típusok, operátorok és kifejezések

Változónevek, típusok

- Változó nevek
  - Első karakter betű
  - Első 31
  - Kis- és nagybetű
  - Kulcsszavak
  - Beszélő nevek, belső, külső változó
  - Megjegyzés /*..*/ , //
- **Alap adattípusok**
  - **char** (1)
  - **int** (!)
  - **float** (1x, 4, 6-7digit, előjel:1, manti.:23, karakt: 8)
  - **double** (2x, 8,15-6 digit, , előjel:1, manti.:52, karakt.: 11)
  - short/long
  - signed/unsigned
  - long double (8, MS -> =double)

Típuskonverziók

- Jobbérték, balérték
  int i;
  i = 5;
- Kisebbről nagyobbra
  char c=5;
  int i;
  double d;
  i = c; ?????
  d = i;
- Nagyobbról kisebbre
  c = i;
  i = d;

Típuskonverziók

- Automatikus konverziók
  - char, short->int
  - float->double
- A nyelv mindig a bal oldal, a cél terület típusára konvertálja a jobb oldal típusát (K&R)
- A kiértékelés során (végrehajtás előtt) a kifejezést alkotó minden elem a „legerősebb” típusra konvertálódik (Pethő)
- char/int, signed / unsigned ? Cmp? -> pl.
  int i = -3;
  printf(" %d %u", i, i);
  -3, 65533

Állandók

- int: 255, 0377, 0xFF
- long: 255L
- float,double:15.75E2 -> 1572, .123
- unsigned int: 255u
- Karakter: ‘a’
- Escape szekvenciák:
- \a bell
- \b back space
- \f from feed
- \r car.ret.
- \t hor.tab.
- \v ver.tab
- \n newline

\’ ’
\” ”
\\ backlash
\ddd okt. konst
\xddd hex. Konst
\0

Ami gyakran keveredik, string konstans

„Hello”

’H’	’E’	’L’	’L’	’O’	’ \0’

’A’ != ”A”

1 bájt ! 2 bájt !

Definíció, deklaráció

Használat előtt mindig!

int x,y,z;

definició != deklaráció

Operátorok, aritmetikai

- **Aritmetikai**
  -, +, /, *, %
  10 / 3 -> 3
  10 % 3 -> 1
- Relációs és logikai
  <,>,<=,>=,==,!=, &&, ||, !

  Kifejezés értéke:
  Hamis: 0
  Igaz:1

  printf("ELSŐ: %d \nMÁSODIK %d ", 3==3, 3==4)

  ELSŐ: 1
  MÁSODIK: 0

Operátorok, inkrementáló

- **Inkrementáló és dekrementáló**

Prefix: ++n, --n x = ++n; n = n + 1; x = n;

Postfix: n++, n-- x = n++; x = n; n = n + 1;

```
int a,b,c;
c = a+++++b; NOK
c = a++ + ++b;

--(c * 3);  NOK

--c++;  NOK
```

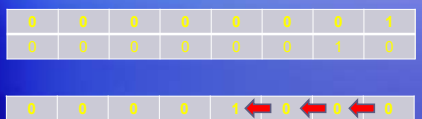

Példa

```
void strcat( char s[], char t[])
{
    int i=0, j=0;

    while( s[i] != '\0')
      i++;

    while ( (s[i++] = t[j++]) != '\0' )
      ;
}
```


Operátorok, bitenkénti

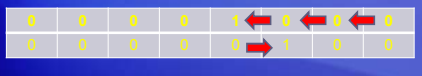
- **Bitenkénti logikai operátorok**

&, |, ^ (kizáró vagy), ~, <<, >>



A & B = 0

A && B = 1



A << 3

A >> 1

- **Értékadó operátorok és kifejezések**

a = a + b -> a+=b, a >>=b, stb.

```
while( (c = getchar()) != EOF )
{

}
```


Operátorok, feltételes

- **Feltételes kifejezés**

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

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

```
z = ( a < b ) ? a : b;
```

- **Objektum méretét meghatározó operátor**

```
double x;
char t[] = "narancs";
char * p = t;

printf("%d-%d-%d-%d", sizeof(double), sizeof(x), sizeof(t), sizeof(p) );
```

8-8-8-4

Operátorok, típus konvertáló

- **Típus konvertáló operátor**

Típuskonverzió kikényszerítése

```
int j = 3;
double k;
k = pow(2.0, (double) j);
```


Operátorok

() [] -> .	bj
(típus) sizeof & * ++ -- ~ !	jb
* / %	bj
+ -	bj
<< >>	bj
< <= > >=	bj
== !=	bj
&	bj
^	bj
|	bj
&&	bj
||	jb
?:	jb
= += -=	bj
,	

Példa

```
int m=3, n=4;
float x=2.5, y=1.0;

x = x + y = y - m
```

x = x + (y=(y-m))
2,5 + (1.0-3) = 0,5

```
int j = 0, m = 1, n = -1

m += ++j * 2;
```

m = m + ((++j) * 2;
1 + (1*2) = 3

Példa

```
while(  (c = getchar()) != EOF )
{
    ....
}
```

```
while( c=getchar() != EOF )
{
...
}
```

?

c = ?

3. rész

Vezérlési szerkezetek

Vezérlési szerkezetek, utasítások

- Utasítások és blokkok

```
;
{}
```

```
{
    double z;
    .
    for (…)
    {
        int z;
        .
        z++;      -> int z!!!
    }
    z++;          -> double z!!!
}
```


if utasítás

- **if utasítás**

```
if ( kifejezés )
    utasítás1;                kifejezés != 0
[else
    utasítás2;]               kifejezés == 0
```

- utasításblokk

```
{
    Utasítás 1;
    Utasítás 2;
    .
    .
    .
    Utasítás n;
}
```


else if

```
if( kifejezés1)
    utasítás1;
else
{
    if(kifejezés2)
        utasítás2;
    else
    {
        if(kifejezés3)
            utasítás3;
        else
            utasítás4;
    }
}
```

```
if( kifejezés1 )
        utasítás1;
else if(kifejezés2)
        utasítás2;
else if(kifejezés3)
        utasítás3;
else
        utasítás4;
```

Switch

Többirányú program elágaztatás

```
int ch;
switch (ch)
{
   case 't':
            utasítás1;
            break;
   case 'h':
            utasítás2;
            break;
   case 'K':
   case 'k':
            utasítás3;          !!!!!!!
   default:
            utasítás4;
}
```


while, for

```
while( kifejezés )
   utasítás;

do
   utasítás;
while( kifejezés )
```

```
kifejezés1;
while( kifejezés2 )
{
     utasítás
     kifejezés3;
}
```

```
for( kifejezés1; kifejezés2; kifejezés3 )
        utasítás;
```


Példa

```
void setmem(char s[], int h, char ch)
{
     while( h )
     {
          s[--h] = ch;
     }//while

} //setmem
```

```
while(a==b)
    !!!!!
while (a=b)
```


break, continue, goto

- **break**
  while, for, do while, switch
- **continue**
  while, for, do while
- goto
  goto címke; ?

4. rész

Függvények és programstruktúrák

Függvények

- Max. 50 sor
- Fv-ben nem lehet másik fv-t definiálni
- {} - blokk
- src.c/src.cpp - modul
- Definíció

```
[ érv.tart ] [ típus ] név ( [ paraméter lista ] )
{
     [ deklarációk ]
     [ definíciók ]
     utasítások;
     [ return [ (v.érték) ] ]
}
```

Fv. deklaráció

Minden függvényt deklarálni **KELL** (C-ben nem kötelező !) használat előtt, azaz meg kell adni a függvény prototípusát (kézjegy).

Később ismét !

Függvény definíció és deklaráció

deklaráció

src1

```
int main(void)
{
    extern int f1(void);
    extern int f2(void);
    ....
    f1();
    ....
    f2();
    ....
    return(0);
}
```

definíció

src2

```
int f1()
{
    ....
    return(0);
}
```

definíció

src3

```
int f2()
{
    ....
    return(0);
}
```


Érvényességi tartomány

src1

```
extern int f1(void);
extern int x1;
int f2(double d);
int x2;
int main(void)
{
    extern int f3(void);
    extern int x3;
    int x4;
    …
}
int x3;

int f2(double d)
{
    int x5;
    …
}
```

src2

```
int x1;
int f1(void)
{
    int x6;
    ....
}

int f3()
{
    extern int x2;
    int x7;
    ....
}
```

x1,x2,x3 : külső, globális
x4,x5,x6,x7 : belső, automatikus (auto)

Static

src2

```
static int y1;

static int f5(void)
{
    static int y3;
    ....
}
```


Regiszter, inicializálás

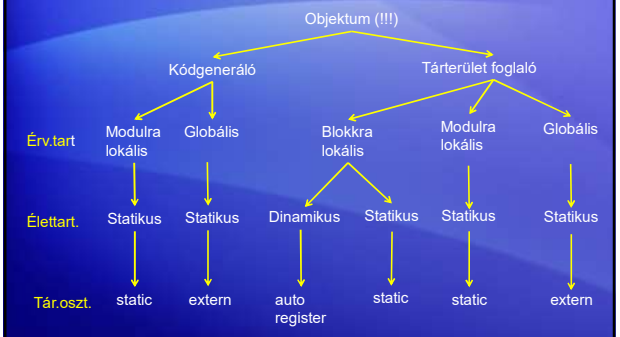
- **Regiszter változó**
  - HW, gyors, MS: esi, edi
  - Automatikus, vagy fv. Paraméterek
  - Nincs garancia

```
register int i;
```

- **Inicializálás**
  - Külső, static –> 0
  - Register, auto -> ?

```
char t[4] = {'e', 'p', 'e', 'r' };
char t[] = "eper";    ???
```


Tárolási osztályok és érvényességi tartomány



Előfeldolgozó

Megjegyzés, /*…*/, //
#include <alma.h>, #include "alma.h"
#define
#undef
#ifdef
#else
#endif
#ifndef
__DATE__
__FILE__
__LINE__
__STDC_
__TIME__
__cplusplus
_DEBUG
….

Mystring.h

Stdio.h (MS2010)

```
#ifndef MYSTRING_H
#define MYSTRING_H
#ifndef NULL
#ifdef __cplusplus
#define NULL   0
#else
#define NULL   ((void *)0)
#endif
#endif
#define MAXBUF  512
```

/* pointer value */

Makrók

```
#define YES    1
#define NO     0
#define MSG    "Haha!"

printf("%d - %s", YES, MSG);

#define MAX(a,b) ((a)>(b) ? (a) : (b) )

#define MIN(a,b) ((a)<(b) ? (a) : (b) )
```


Példa

Stdio.h (MS2010)

```
#define getchar()       getc(stdin)
#define putchar(_c)     putc((_c),stdout)

#define getc(_stream)   fgetc(_stream)

#define fgetc(_stream)    (--(_stream)->_cnt >= 0 \
        ? 0xff & *(_stream)->_ptr++ : _filbuf(_stream))

#define putc(_c,_stream)  (--(_stream)->_cnt >= 0 \
        ? 0xff & (*(_stream)->_ptr++ = (char)(_c)) :  _flsbuf((_c),(_stream)))
```


Példa

```
#define negyzet(x)              x*x
i = negyzet(x+3);               i = x + 3  * x + 3
#define negyzet(x)              (x) * (x)
#define negyzet(x)              (x) * (x);
y = negyzet(x) + 2 ;            x * x; + 2;
#define limit_number(a)         ((a>1000) ? 1000 : (a) )
a = limit_number( a++ );        ((a++ > 1000) ? 1000 : a++ )
```

Makró ? Függvény

5. Rész

Mutatók és tömbök

Mutató fogalma

A mutató olyan változó, amely egy objektum címét tartalmazza, így a mutatón keresztül az objektum elérése indirekt módon lehetséges.

```
int x, * xp;

xp = &x;

*xp = 5;        5->x !
```

kifejezés
= & konstans
reg.változó

124	0
123	05
122	0
121	0
120	7B

x -> 123

xp -> 120

(123 -> 0x7B)

NULL
vagy
egy változó címe

Mutatót használat előtt inicializálni KELL!

Tömbök

```
int t[3];                INDEX : 0,1,2   INDEX ELL.!!!!
int *p, k;
p = &t[0];               -> *p = t[0];
k = *p + 1;              -> k = t[0] + 1;
k = *(p+1);              -> k = t[1];

p = p + 1;               növekmény mértéke?

Fordító tömbnév -> mutató

p = &t[0];       ==      p = t;
```

DE!

Mutatók és fv.argumentumok

swap(int x, int y) x < -- > y

Érték szerinti hívás.
Globális változók?
Változó címe, cím szerinti paraméterátadás

```
...
int x=5, y=10;
swap( &x,&y);
...
int swap(int * px, int * py)
{
    int tmp;
    tmp = *px;
    *px = * py;     NEM: px = py !!!!
    *py = tmp;
}//swap
```


Címaritmetika

Műveletek:
+, - (csak egész)
cmp, p1-p2 (azonos tömb)
Inicializálás: objektum címe, NULL

```
int t[10];
int *p, k;
p = &t[5];

k = *p++;

k = (*p)++;

k = ++*p;

k = *++p;
```

```
*++a+=*b++**++c;

*++a += *b++ * *++c;
```


Példa

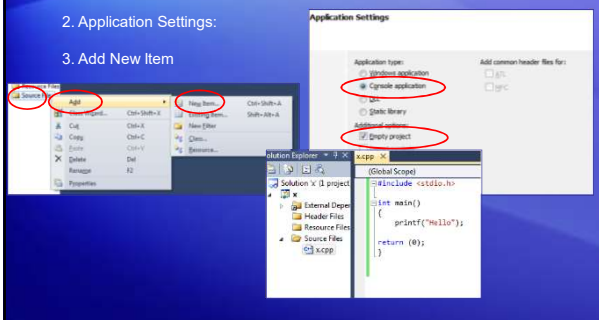
```
int strlen(char * s)
{
    char *p=s;
    while(*p)
    {
        p++;
    }//while
    return(p-s);
}//strlen
```

```
char * index(char ch, char * str)
{
    while( *str )
    {
        if(*str++ == ch)
            return(--str);
    }//while
    return(NULL);
}//index
```

```
void str(char * s1, int hossz, char * s2)
{
    while( (hossz--) && (*s1++ = *s2++) )
    ;
}//str
```


MS Visual Studio

1. New Project\Visual C++ \ Win32\ Win32 Console Application
2. Application Settings:
3. Add New Item



És még egy apróság…

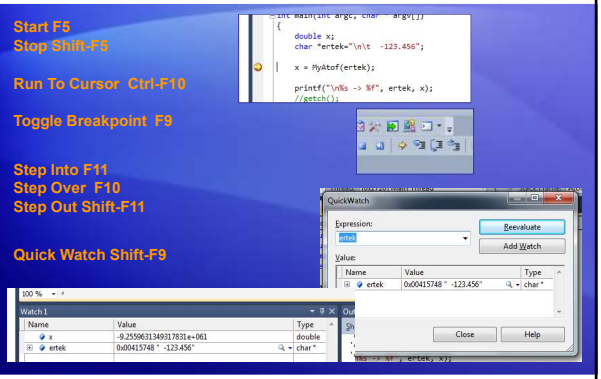
Start F5
Stop Shift-F5

Run To Cursor Ctrl-F10

Toggle Breakpoint F9

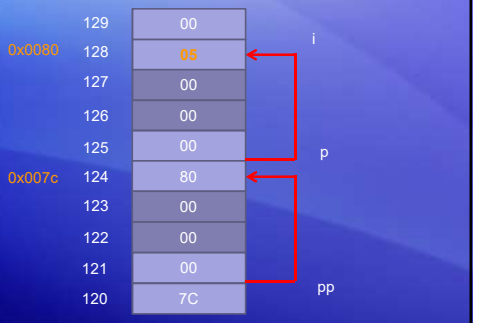
Step Into F11
Step Over F10
Step Out Shift-F11

Quick Watch Shift-F9



Mutatókat megcímző mutatók

```
int i=5;
int * p;
int  ** pp;
…
p = &i;
pp = &p;
…

*pp -> p

*p -> i
```



Kétdimenziós tömbök

```
char t[3][6]={ "dió", "alma", "eper" };
```

'd'	'i'	'ó'	'\0'	'\0'	'\0'
'a'	'l'	'm'	'a'	'\0'	'\0'
'e'	'p'	'e'	'r'	'\0'	'\0'

```
char * pt[3]={ "dió", "alma", "eper" };
```

1000	'd'	'i'	'ó'	'\0'	
1004	'a'	'l'	'm'	'a'	'\0'
1009	'e'	'p'	'e'	'r'	'\0'

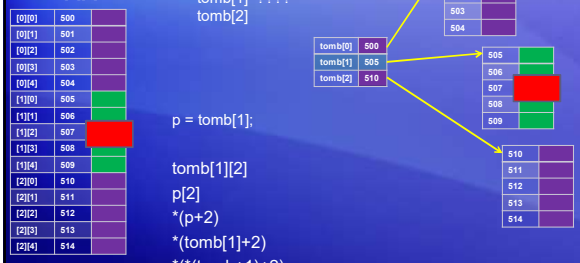
MSVS : char * p = „alma" -> const char * p !

Kétdimenziós tömbök…

A nyelvben kétdimenziós tömb?

```
int * p;
int tomb[3][5];
```

tomb[0]
tomb[1] ????
tomb[2]




```
p = tomb[1];
```

tomb[1][2]
p[2]
*(p+2)
*(tomb[1]+2)
((tomb+1)+2)

Parancssor argumentumok

```
c_ppt alma körte eper

argc  = 4
argv:
```


```
int main(int argc, char *argv[], char *envp[])
{
  char ** q;
  printf("argc: %d\nargv[0]: %s\nArgumentum lista: \n",
          argc, argv[0] );

  while(--argc>0)
  {
     printf( (argc>1) ? "%s " : "%s\n", *++argv);
  }

  printf( "Környezeti változók\n");
  for(q=envp; *q; q++)
  {
     printf("%s\n", *q);
  }
  return(0);
}
```

```
argc: 4
argv[0]: c_ppt
Argumentum lista:
alma körte eper
Környezeti változók
ALLUSERSPROFILE=C:\ProgramData
APPDATA=C:\Users\vkoschek\AppData\Roaming
asl.log=Destination=file
CLASSPATH=.;C:\Program Files\QuickTime\QTSystem\QTJava.zip
COMMANDER_DRIVE=C:
COMMANDER_INI=C:\Users\vkoschek\AppData\Roaming\GHISLER\wincmd.ini
COMMANDER_PATH=C:\Program Files\totalcmd
CommonProgramFiles=C:\Program Files\Common Files
…..
```


Függvényeket megcímző mutatók

```
int i;
int sub0(), sub1(), sub2();
int (*f)();
int ( *ftomb[3] )() = {sub0, sub1, sub2};
f = sub0;
(*f)();
for(i=0; i<3; i++)
{
    ( *ftomb[i] )();
}
```

*ftomb[3]() != (*ftomb[3])()

Példa

```
int parse( char *ibuf,  *obuf )
{
   for (i=0; i<NUM_ROUTINES; i++)
      {
      j=strlen(cmd_code[i]);
      if (strncmp(ibuf, cmd_code[i], j) == 0x00) break;
      }
….
      return(cmd_routine[i](ibuf, obuf, strlen(ibuf)));
}

  int (*cmd_routine[])() = {  cmd_online, cmd_read, cmd_remove,
  cmd_maxid, cmd_time,  cmd_dcount, cmd_ttune, cmd_mcard,
  cmd_pwd,  cmd_bl, cmd_vol, cmd_com1, cmd_com3, cmd_bcode,
  cmd_reader1, cmd_reader2, cmd_tmstp, cmd_idstp, cmd_srstp,
  cmd_msg };
```

```
char *cmd_code[] = {
      "ONLINE",     /* inquire on-line stations */
      "READ",
     "REMOVE",    /* access data file */
      "MAXID",     /* maximum station ID */
      "TIME",      /* system time */
      "DCOUNT",   /* number of data in buffer */
      "TTUNE",     /* calendar tuning */
      "MCARD",    /* manager card */
      "PWD",       /* manager password */
      "BL",        /* backlight */
      "VOL",       /* buzzer volume */
      "COM1",      /* com 1 */
      "COM3",      /* com 3 */
      "BCODE",    /* symbology */
      "RDR1",      /* reader 1 */
      "RDR2",      /* reader 2 */
      "TMSTP",    /* time stamp */
      "IDSTP",     /* station ID stamp */
      "SRSTP",    /* source stamp */
      "MSG"        /* message period */};
```

```
/*****************************************************************************/
/*  <DCOUNT> data count                                                      */
/*****************************************************************************/
int cmd_dcount(char * ibuf,  char * obuf,  int cmd_leng)
{
   sprintf(obuf, "%lu\r", data_count);
   return(COMPLETE);
}
```

Típusnév definíciók

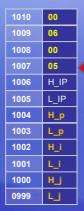
- Új adattípus
- Gépfüggőségi problémák
- Dokumentálás
- Komplex típusok

#define ? typedef

```
typedef  int EGESZ;
…
EGESZ  x1, x2;
```


Változó argumentumszámú függvények

```
int x=5, y=6;
f(x,y);

push y
push x
call _f
add sp, 4

int f( int j, int k )
{
    char * p = &j;
    int i, j;
    …
}
```



&j

y = ?

- azonos típus
- string

6. rész

Struktúrák

Struktúra

Több, esetleg különböző típusú változó együttese, amelyeket az egyszerű kezelhetőség érdekében egy név alatt fogunk össze egy adatszerkezetbe.

```
struct struktúranév
{
        Struktúra tagok;
} struktúra változó1, vált2;
```

```
struct cardstruct{
char * street;
long street_num;
…
} card1;
```

- Hivatkozás: card1.street_num

```
struct cardstruct card2, card3;
```

- Inicializálás: struct card1 = { „Váci”, 1,…};
- Műveletek:
  - &, címének előállítása
  - Hozzáférés az adattagokhoz
  - sizeof
  - Értékadás: card1 = card2;

Struktúra

- Struktúra ptr

```
struct cardstruct  s;
struct cardstruct *p = &s;
```

- p++
- Mint paraméter
- Használat
  - p->street_num
  - (*p).street_num
  - s.street_num
  - (&s)->street_num
- Önhivatkozó struktúrák, saját megnevezését nem, de mutatót igen.

```
struct  personstruct{
char name[20];
struct personstruct * next;
…. }
```

- typedef struct personstruct PERSON;

Bitmező

Olyan adatstruktúra, mely lehetővé teszi egyes objektumok bitenkénti elérését. Általában int méretű.

```
struct screen
{
        unsigned character: 8;
        unsigned fgcolor:3;
        unsigned intensity:1;
        unsigned bgcolor:3;
        unsigned blink:1;
} screenbuf[25][80];

….

screenbuf[13][53].blink = 1;
```

Union

Az union olyan változó, amely (különböző időpontokban) különféle típusú és méretű objektumokat tartalmaz.

```
union u_sample
{
        char c_val;
        int i_val;
        long l_val;
} u;

u.c_val = '0';
```


Union 2.

```
struct WORDREGS { unsigned int ax, bx, cx, dx, si, di, cflag, flags; };

struct BYTEREGS { unsigned char al, ah, bl, bh, cl, ch, dl, dh; };

union REGS
{
        struct WORDREGS x;
        struct BYTEREGS h;
} r;
```

Borland C/C++ bios.h

```
r.x.ax = 0x1234;

r.h.al  -> 0x34
r.h.ah -> 0x12
```


Felsorolt típus

- A **felsorolásos típus** egész értékek egy diszkrét halmazának a leírására szolgál.

```
enum napok { hetfo, kedd, szerda, csutortok, pentek, szombat, vasarnap };

        hetfo -> 0
```

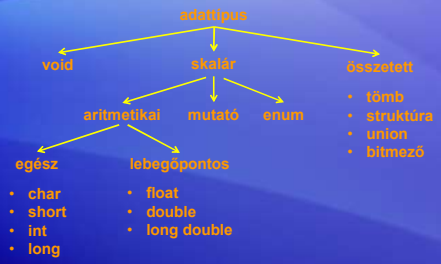
- void típus

```
        void f();
        void * p;
```

- const módosító jelző

```
        const double pi = 3.1415926;
```


Adattípusok áttekintése



Köszönöm a figyelmet!

koschek.vilmos@nik.uni-obuda.hu