C++ ↔ C

C-ben:

- nincs bool típus:
  - hamis: =0
  - igaz: ≠0
- for-ban nem lehet a ciklusváltozót deklarálni (c99-től lehet csak)
- nincsenek osztályok:
  - nincs stream osztály (<< és >> operátorok, …)
  - nincs string osztály (string típus, = operátor, …)

hellow.c

```
#include <stdio.h>
int main()
{
   printf("Hello World!\n");
}
```

Fordítás és futtatás

- Fordítás:
  - gcc forrás.c → a.out
  - gcc forrás.c -o cél vagy –océl → cél
- Futtatás:
  - ./cél
  - vagy PATH környezeti változó elejére hozzáadni a . elérési utat (.profile)

Konzolra írás

<stdio.h>

- **printf**({formátumstring}[, {változólista}])

escape szekvenciák:

- **\n – új sor**
- **\t - tabulátor**
- **\” – idézőjel**
- **\’ – aposztróf**
- **\0 – null karakter**
- **\r – kurzor a sor elejére**
- **\\ – backslash**

formátumleírók:

- **%i – egész**
- **%f – valós**
- **%c – karakter**
- **%s – string**

string

- null-terminált karaktersorozat:

0	1	2	3	4	5	6	7	8	...
H	a	j	r	á	␣	F	r	a	...

str[5] = '\0' ≡ str[5] = 0;

0	1	2	3	4	5	6	7	8	...
H	a	j	r	á	'\0'	F	r	a	...

- deklaráció (statikus):
  - char str[80];
  - char str[80]="Hajrá Honvéd!";
  - char str[]="Hajrá Vasas!";

Mutató

* - „értéke” operátor

& - „címe” operátor

mutató típusú változó:	char *ptr_str;
mutató értékadás:	ptr_str = &str[0]
	ptr_str = str;
mutató hivatkozás:	ptr_str &ptr_str[0]
	ptr_str+6 &ptr_str[6]
elemre hivatkozás:	ptr_str[0] *ptr_str
	ptr_str[11]
	*(ptr_str+11)

mutató példa

```
#include <stdio.h>

int main()
{
        char s1[] = "Forras szoveg";
        char *s2, *s3;
        s2 = s1;
        s3 = &s1[0];
        printf("s1=%s, s2=%s, s3=%s\n",s1,s2,s3);

        s2 = s1+7;
        s3 = &s1[7];
        printf("s1=%s, s2=%s, s3=%s\n",s1,s2,s3);
}
```

Dinamikus memóriafoglalás

<stdlib.h>

- {mutató típusú változó} =
  [({elemtípus}*)]**malloc**({memóriaméret})
  Lefoglal {memóriaméret} méretű memóriaterületet.
  Értéke: A lefoglalt memóriatartomány kezdőcíme, vagy 0 (hiba)

- **free**({mutató})
  Felszabadítja a {mutató} kezdőcímű foglalt memóriaterületet.

Dinamikus memóriafoglalás

<stdlib.h>

- {mutató típusú változó} = [({elemtípus}*)]**realloc**({mutató},{memóriaméret})

  Módosítja a {mutató} által meghatározott foglalt memória méretét {memóriaméret} méretűre, az eredeti tartalom értelemszerű megtartásával.

  Értéke: A lefoglalt memóriatartomány új kezdőcíme, vagy 0 (hiba).

Feladat

Hozzuk létre dinamikusan str stringet, mely legfeljebb 10 karakter hosszú szöveg tárolására képes!

```
char *str = (char*)malloc(11*sizeof(char));
```

```
char *str = malloc(11);
```

Szabadítsuk fel az str string által lefoglalt memóriaterületet!

```
free(str);
```

String értékadás

<string.h>

- **strcpy**({cél mutató}, {forrás mutató | konstans})

```
#include <stdio.h>
#include <string.h>
int main()
{
        char s1[20],s2[20],s3[20];
        strcpy(s1, "STRCPY forras");
        strcpy(s2, s1);
        strcpy(s3, &s1[0]);
        printf("s1=%s, s2=%s, s3=%s\n",s1,s2,s3);
}
```

További string függvények

<string.h>

- **strlen**({mutató})
  Értéke: A {mutató} által meghatározott string hossza.
- **strcmp**({mutató1}, {mutató2})
  A két {mutató} által mutatott stringek tartalmát hasonlítja össze.
  Értéke: negatív, ha {mutató1}<{mutató2}
  0, ha {mutató1}={mutató2}
  pozitív, ha {mutató1}>{mutató2}
- **strcat**({cél mutató}, {forrás mutató} | {konstans})
  Hozzáfűzi a {cél mutató} által meghatározott stringhez a {forrás mutató} által meghatározott string tartalmát, vagy a {konstans} stringet.

Olvasás konzolról

<stdio.h>

- **scanf**({formátumstring}, {változó-kezdőcím lista})

formátumleírók:

- `%i` – egész
- `%f` – valós
- `%c` – karakter
- `%s` – string
- `%s{méret}` – legfeljebb {méret} karakter hosszúságú string

Feladat

Olvastassunk be a konzolról egy legfeljebb 20 karakter hosszúságú szöveget, majd írassuk ki amit beolvastunk!

```
#include <stdio.h>
int main()
{
        char str[21];
        scanf("%s20",str);
        printf("%s",str);
}
```

Feladat

Olvastassunk be a konzolról egy egész és egy valós számot egyetlen függvényhívással, majd szintén egyetlen függvényhívással írassuk ki amit beolvastunk az alábbi formában:

Egész: 5
Valós: 3.14

```
int egesz;
float valos;
scanf("%i%f", &egesz, &valos);
printf("Egész: %i\nValós: %f\n",egesz,valos);
```

Feladat

Csináljunk length függvényt!

- hívás: hossz = length(str);
- értéke: a string hossza (strlen)

length függvény

```
int length(char *str)
{
        int len = 0;
        while (str[len] != 0)
                len++;
        return len;
}
```

length függvény

Használjuk ki, hogy a paraméter egy mutató!

```
int length(char *pointer)
{
        int len = 0;
        while (*pointer != 0)
        {
                pointer++;
                len++;
        }
        return len;
}
```

length függvény

A mutató léptetését a ciklusfeltételben is elvégezhetjük.

```
int length(char *pointer)
{
        int len = 0;
        while (*pointer++ != 0)
                len++;
        return len;
}
```

length függvény

Mivel az =0 a hamis a !=0 pedig az igaz, optimalizáljuk a ciklusfeltételt.

```
int length(char *pointer)
{
        int len = 0;
        while (*pointer++)
                len++;
        return len;
}
```

length függvény

Ha elmentjük a kezdő pointert, nem kell a len változó.

```
int length(char *pointer)
{
        char *start = pointer;
        while (*pointer++);
        return --pointer-start;
}
```

struktúra

típus definíció:

```
struct {típusnév}
{
        {mezőtípus1} {mezőnév1};
        …
        {mezőtípusN} {mezőnévN};
};
```

struktúra

változó deklaráció:

struct {típusnév} {azonosító}

mezőhivatkozás: {azonosító}.{mezőnév}

pointer változó deklaráció:

struct {típusnév} *{pointerazonosító}

mezőhivatkozás:

(*{pointerazonosító}).{mezőnév}

{pointerazonosító}->{mezőnév}