C++, mint egy jobb C nyelv

1. néhány különbség:
    a. függvény: int fv(…) →
    b. main fgv. :
        i. int main() →
        ii. int main(int argc, char* argv[])

2. változó deklaráció mint utasítás
    a. int a = fv(…); (bárhol lehet deklarálni) → figyelni kell hogy hol érvényes a változónk: deklarálástól az utolsó bezáró kapcsos zárójelig…

3. függvénynevek túlterhelése

```
struct ido
{
        Int ora;
        Int perc;
};

Void ido_kiir(struct ido* t) {…}
Void ido_kiir(int ora, int perc) {…}
Void ido_kiir() {*aktualis ido kiirás*}

//C++-ban ez nem hiba!! Különböző paraméterlista különbözteti meg egymástól őket
És a visszatérési érték NEM számít

int cppfunc()
int cppfunc(int a, double b)
void cppfunc() → ez már hiba!! Mert csak visszatérési értékben van külöbség
```

(C-ből C++-ba)

```
/C-ben/
void c_osszead(int a, int b)
{
        return a+b;
}

/c++ban/
#include <stdio.h>

extern „c” c_osszead(int, int);   //ezzel megmondjuk, hogy c fájlból linkelje!

int main()
{
        printf(„%d”, osszead(2,3);
```

(C++-ból C-be)

```
//C++
int cpposszead(int a, int b)
{
        return a+b;
}

extern „c” int c_osszead(int a, int b)
{
        return cpposszead(a, b);
}

//C program
int c_osszead(int, int);

void main()
{
        printf(„%d”, c_osszead(2,3));
        return 0;
}
```

Inline fgvenyek

```
int max(int a, int b)
{
        return (a>b) ? a : b;
}
```

```
//C
#define MAX(a,b) ((a)>(b)) ? (a) : (b);
…
int x = 1; int y = 0;
```

MAX(x++, y); → 2-szer fogja növelni x-et!

```
//C++

inline int max(int a, int b)                    //kiküszöböli a makrót!
{
        return (a>b) ? a : b;
}
```

inline-t NEM biztos hogy a fordító teljesíti, ezeknél NEM teljesíti:

- rekurziónál
- cimkére ugrás
- függvény pointer
- egyéb…

4. Konstansok

//C

#define MAX 5

//C++

const int max = 5;

- kötelező inicializálni
- teljesen más típus a compiler számára mint az int ~~ unsigned-ra hasonlít
- nem lehet változtatni
- nem lehet nem konstanssá konvertálni automatikusan

int fv(int i) {…};

fv(max); //HIBA!
fv((int)max); //OK

<u>pointereknél</u>

két dolog lehet konstans:
- mutatott érték
- mutató értéke

const char * ptr;

- *ptr = 'a'; //HIBA
- ptr++; //OK

char * const ptr = …;

- *ptr = 'a'; //OK
- ptr++; //HIBA

<u>konstans referencia</u>

struct szemely
{
…
};

void fv(const struct személy &sz)
{
…//nem változtathatom meg az értéket → jó mert nem megy le a stackre! gyorsabb!
}

```
struct szemely init.szemely(…)
{
        …
}

…

fv(init_szemely(„Árpád”,…));
```

5. Alapértelmezett függvényargumentummal

```
double circumference(int x, int y = 10, double r = 3.14) {…}
```

circumference(3); ⇔ circumference (3 , 10 , 3.14);
circumference(3,12); ⇔ circumference(3, 12, 3.14);

- csak jobbról balra!
- jobbra a default értékek

példa:

```
int f1(int a) { return a;}

int &f2(int a) {return a;}

int &f3(int &a) {return a;}

int f4(int &a) {return a;}
```

a.) melyik hibás, jóllehet lefordul?

*futási hiba: f2, mert stackben érvényes a refernciája

b.)melyik hívható önmaga paramétereként?