# 3. előadás: Nyelvi fordítás, tokenizing

## Teszt

A "gyanús" honlapokon levő tesztkérdések közül csak az marad benne a tesztben, ami hibás. Így kerüljük el a csalást.

## Compiler milyen token​ekre bontja a kódunkat

int main**(){**

std::cout**<<**"Hello World!"**<<**std::endl**;**

**return** 0**;**

**}**

* Kulcsszavak, pl.: int, **return**, ezeket nem használhatjuk azonosítóként.
* Azonosítók, pl.: main, std, cout
* Konstansok / Konstans literálok, pl.: 0
* Konstans szöveg​literálok / string​ literálok, pl.: "Hello World!"
* Operátorok, pl.: :: (scope), **<<**
* Szeparátorok, pl.: **{**, **}**
* Vannak továbbá a nem egyértelműek, pl.: a vessző

## Kulcsszavak

Összes kulcsszó csupa kisbetű. A nyelv alap építőelemei. Jelentéssel bírnak. Nem használhatjuk változónévként.

* auto
* if, while
* int, double
* class

## Azonosítók

Betűvel kell kezdődniük. Betűvel vagy számmal folytatódik. Nincs benne whitespace, bár az egyik token​-en belül sem lehet. Az aláhúzás (\_) itt betűnek számít.

Case sensitive. Valid azonosító például az (nagy i kis f). Különböző de valid az . Valid azonosító az is. De ne válasszunk ilyen változóneveket! (Névkonvenciókkal nem foglalkozunk, de fontos.)

Azonosító nem lehet kulcsszó.

### Névkonvenciók

JAVA névkonvenciója a Camel​Case. példa változónév:

Létezett régen a Hungarian Notation. BCPL-ben az volt a divat, mert a gépi szó volt az egyetlen változó típus. A változó neve elé "beprefixálták" a típusát:

* : boolean vagy flag
* : Long pointer egy zero-terminated string​-re (Nem ?)

C++-ra nem jellemző egy standard, mindenki olyat használ, amit szeretne. STD-ben például a szóhatároknál aláhúzás: , De használjuk egységesen. Ne össze-vissza.

Továbbá a map, set multi​map, multi​set mögötti implementáció használ dupla aláhúzással kezdődő azonosítókat. Pl.: Ha mi is ilyen stílusú változónevekkel dolgozunk, névütközéseket produkálhatunk. Ezért ne kezdjük változóneveinket dupla aláhúzással.

## Konstans szöveg literál

Minden konstansnak van típusa és értéke.

Mi a "Hello" típusa? Nem , hanem .

Benne lesz a tárgykódban. Látszik is alacsony szinten.

Miért pont ? Mert tartalmazza a lezáró karaktert. **{**'H'**,** 'e'**,** 'l'**,** 'l'**,** 'o'**,** '\0'**}**

Miért char**\***? Szabad-e ilyet mondani: char**\*** p **=** "Hello"**;**? Lehet, de nem szabad. Fordításkor figyelmeztetés: Deprecated conversion. const char**\*** p **=** "Hello"**;** Ez oké. p**[**1**]=**'a'**;** Ez nem oké. Futás idejű hiba: Segmentation fault. Konstansoknak fenntartott tárterület írása nem megengedett. Ha futás közben módosítható string​et szeretnénk:

Érdekesség: parancsait át lehetett írogatni szövegszerkesztővel, csak a szöveghossz kell, stimmeljen. A literálok ott vannak a kódban.

## Konstansok

Van értéke és típusa.

14 típusa: int Előjeles egész.  
Értéke: 14

Mi is az az int? A processzor számára minden adat 1-esekből és 0-kból álló valami. A típusok programozói absztrakciók. Bizonyos bitsorozatokat leszűkítünk, hogy hogyan használjuk.

A processzornak nincs olyanja, hogy int. 32 vagy 64 bitet mi kezelünk egy egységként. (Régen 8 v 16.)

Azért működik ma is az 1970-ben írt C kód, mert így definiálták az integert: Legyen az adott gép leghatékonyabb számábrázolása egész típusra. JAVA-ban épp ellenkezőleg: Mindenhol 32 bites az integer. Amíg világ a világ.

### Short, long

Mindig igaz állítás: **sizeof(**short**)<=sizeof(**int**)<=sizeof(**long int**)**. Pl.: 16 bites gépen 16 bites volt az integer. (1 előjelbit, a legnagyobb ábrázolható egész szám.)

### Túlcsordulás

int x**;**

20000**+**x**;** //**(**int**)** Túlcsordul, ha x is 20000-es nagyságrendben van.

20000L**+**x**;** //**(**long**)** Nem csordul túl.

### Előjel nélküli

unsigned int**;**

20U

Felszabadul 1 bit.

Előjel nélküli long: unsigned long

20LU vagy 20UL

### Méret

**sizeof(**T**)==sizeof(**unsigned T**)==sizeof(**signed T**)**

T – template paraméter (Később lesz.)

### Char

Itt fontos a signed, unsigned.

Mert a karakterábrázolás mindenhol eltért.

Előjelessége implementáció / fordítóprogramtól függő.

### Oktális (nyolcas) számrendszer

Nullával kezdődő szám.

int 014 értéke 12.

### Hexadecimális

int 0xA4**;** //Értéke: 164

Ismétlünk középiskolai anyagot:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Gyakorlat vége.