# 2. gyakorlat

## Ismételtünk

ZH-n nem lehet CodeBlocks, pluszminusz milyen rossz, stb.

Megint felrajzoltuk ezt:



(Mnemonics, gépi utasítás, assembly.)

Fordítás eredménye: Object file, ebből lesz futtatható állomány, linkelés után.

## Makefile

Fordító függő fájl, ami összehangolja a fordítás menetét.

## Fordító kapcsolói

g++ hello.cpp hello2.cpp -o hello.exe -w -wall -werror -ansi -pedantic

Felsoroljuk a forrásfájlokat, szóközzel elválasztva.

Majd kapcsolókat rakunk:

* -o hello.exe (kimenet neve)
* -w (Figyelmeztetéseket (warning) generál)
* -Wall További figyelmeztetéseket kapcsol be.
* -g Debug​golható kódot generál. Breakpoint​okat, tudunk berakni, majd soronként léptethetjük a programot, figyelve a változókat.
* -werror Figyelmeztetések helyett hibákat generál és meg is áll a fordítás, ha talál ilyet.
* -ansi -pedantic
* -I "C:\inc" További include könyvtárak hozzáadása (JAVA-ban class​path)

## Lehetséges hibák

* Fatal error (Kritikus hiba, pl. elfogy a fordító memóriája)
* Compiler error (Szintaktikai, szemantikai)
* Warning (A program lefordul, de valószínűleg hibát fog okozni futás közben.)
	+ Nincs új sor a fájl végén
	+ Előjeles és előjel nélküli (signed, unsigned) számok összehasonlítása
	+ Nem használt változó
	+ Nem használt függvényparaméterek

## Hello World.​c

#include "stdio.h"

int main**(){** //Szabvány szerint ettől a ponttól kezdődik a futtatás. Ma "void main()" már nem használandó. Paraméterlistája lehet változó, általában argc, argv[].

 printf**(**"Hello %s!"**,** név**);** //Globális függvény, include-olni kell hozzá az stdio-t. %s helyett lehet %c, %f, %d, stb, akár többször is.

 **return** 0**;** //Ha hiba történt a futás során, az operációs rendszer így tudja meg. A visszatérési értéket megkapja az operációs rendszer. Egyedül a 0 a jó, minden más hibának számít.

**}**

JAVA-ban a printf: MessageFormat**.**format**(**"Hello {0}!"**,** név**);**

printf helyett van fprintf is, ami $stdout$ helyett máshova is tudott írni, például hibakimenet, fájl.

## Deklaráció

Ahhoz, hogy az x**=**y**+**f**(**z**)** kifejezésnek legyen értelme, meg kell adni, hogy $x$, $y$ és $f$ mik. (Változó vagy függvény, mi a típusuk, $f$ elfogad olyan típusút, van összeadás a két típus között, stb.)

Fölé kell írni, hogy:

int x**,** y**,** z**;**

int f**(**int param1**);**

## Hello World.​cpp

A régi c-s könyvtárak még elérhetőek, de $"stdio.h"$ helyett $<cstdio>$. (Le kell vágni a $.h$-t és elé kell írni, hogy $c$.)

#include <iostream> //Nem a projektkönyvtárhoz relatívan keresi, mint az idézőjeles verzió, hanem speciális include útvonalakon.

**using namespace** std**;** //Névtér nélkül nem működik a cout, csak az std::cout. Ez a névütközések elkerülése miatt része a szabványnak. A legtöbb utasítás az std névtérben van. Így feloldható marad a legtöbb változónk.

int main**(){**

 cout**<<**"Hello "**<<**név**<<**"!"**<<**endl**;**

 //A cout, a **<<** operátor és az endl miatt include-olni kell az iostream-et.

 // A cout így van deklarálva: ostream cout**;**.

 // A **<<** így van deklarálva: ostream **operator<<(**ostream o**,** char**\*** str**);**.

 // Az endl a kiíró utasítás (+ új sor). (Pufferelés (buffering) miatt.) A '\n' is üríti ugyan a buffert, de nem szabványos. Inkább a cout**<<**flush -t használjuk.

 **return** 0**;**

**}**

## Globális változók, namespace

Osztályon, névtéren kívül deklarált változó globális.

Deklaráláskor a változókat mindig egy bizonyos hatókörbe vezetjük. (globális, lokális, stb.)

A c szabvány szerint a változók neve bármilyen hosszú lehet, de a fordítók és a linker​ek 32 karakter után általában levágják.

Névteret globális helyen lehet létrehozni, de namespace​-en belül lehet létrehozni másikat, egymásba ágyazva.

Négy mód a használatra:

* **using namespace** std**;** direktíva. Kiemeli az összes $std$-beli deklarációt globális térbe. ZH-n nem érdemes használni, -1 stílus jegy.
* **using** név**;** direktíva. Ezzel csak egy nevet, például az $std::cout$-ot emelem ki globális névtérbe.
* **::**cout Hatókör-feloldó operátor. (scope) Globális névtérre hivatkozik. Például $int cout$ változónkat így érhetjük el.
* std**::**cout Konkrétan kiírjuk, hol van. Ezt tartják szépnek, ZH-n érdemes így programozni.

Header fájlba szigorúan tilos **using namespace** std**;**-t írni. Elronthatja a cpp fájlokat, amik használnák a header​-t.

## Pluszminusz

Fordítás lépései:

1. [Preprocesszor] ← {Forráskódok + header fájlok}
2. [Fordító] ← {Előfeldolgozott forrásfájl}
3. [Assembler] ← {Assembly kód}
4. [Linker] ← {Gépi kód}
5. Futtatható állomány