# r1. gyakorlat

Gyakorlatvezető: Májer Viktor jokmokk@gmail.com

Ez a tárgy már nem az alapokról szól.

## Időpont

Kezdés: 8:05 :(

Vége: 9:35

## Honlapok

<http://gsd.web.elte.hu/> (<http://aszt.inf.elte.hu/~gsd/pny2/html/>)

<http://people.inf.elte.hu/majer_v/>

## Követelményrendszer

Maximum 3 hiányzás. (Régen nem volt kötelező bejárni a gyakorlatra.)

Pluszminusz lesz, 4 db. +, -, 0 jár rá. Aki pluszban van, az jöhet ZH-zni. Aki mínuszban van az is, de az elméleti ZH pontszámából annyit vonnak le.

ZH: félév végén. CodeBlocks tiltott. Notepad++ engedett. g++-szal kell tudni fordítani.

Elméleti rész: 15 kérdéses kvíz (4 válaszból 1 jót ki kell választani). Minimum 8-ra kell válaszolni, akkor mehetünk a gyakorlati részre.

Gyakorlati rész: Egy fájlhoz kell template-t (típust, adatszerkezetet) írni. Ki kell kommentezni annyit, amennyit meg tudunk oldani. (2-5) A végén meg kell védeni.

## Ajánlott irodalom

Bjarne Stroustrup: A C++ Programozási nyelv (első kötet, 4 fejezettől)

Nyisztor Károly: Gyakorlati C++

Konzultáción múlt évi ZH-kat oldunk meg. Utolsó gyakorlaton is lehet, ha belefér.

# Fordítás menete

Van egy kód:



Ezzel két dolgot lehet kezdeni:

## Interpretálás

A Python interpretált nyelv.

Előny: Rögtön futtatjuk. Ahogy megírtuk a programszöveget, mentés után az interpreter (értelmező) rögtön futtathatja.
Nincs fordításidő. (Ez ma már nem annyira fontos, de régen minden sokáig tartott. Bonyolult programok ma is másfél óráig fordulnak. Majd kirakjuk a hiányzó pontosvesszőt és fordítunk megint másfél óráig.)

Hátrány: Futás közben kell értelmezni az utasításokat. Ez lassú.
Ha újra futtatjuk, megint ugyan olyan lassú. (Van olyan interpreter, ami tud optimalizálni, például ha nem változik a program, de nem mind.)

Előny/hátrány: Hibák későn derülnek ki. Csak akkor dob hibát, ha odáig eljut az interpreter. Ez rossz, ha biztonságra törekszünk, de hasznos is:
Nem kell a program minden részének működnie. Ki lehet próbálni a programot, ha hiányos is.

## Fordítás

A C++ fordít.

Előfeldolgozás: Értelmezzük a kódot futtatás előtt. Fordító / compiler csinálja.

Megtalálja a fordítás közben kiszűrhető hibákat: Szintaktika (hiányzik az endif vagy egy pontosvessző), lexikális hiba (nem támogatott karakter), szemantika (string számként), stb.
Vannak olyan hibák, ami csak futtatás közben derül ki. Például: Null pointer Exception

Fordítási idő nagy lehet. Ma is van olyan nagy projekt, hogy módosítok egy betűt és fordul másfél óráig.

Többször futtatható, csak egyszer kell fordítani.

# C++ fordító

Először szövegszerkesztőbe bevisszük a programot. Nem feltétel a $.cpp$ kiterjesztés, de konvenció. A fejállomány $.h$, "header". Template-knél $.hpp$, de ez csak később. (C-ben csak $.c$ a kiterjesztés.)

Ha jól van megírva a program (Jól include-oltuk a fejállományokat), a fordítónak csak cpp fájlokat kell átadni.

## Preprocessor/precompiler (előfeldolgozó)

Mielőtt eljutna a fordítóhoz a program…

Forrás- és fejfájlokat adjuk neki.

A preprocesszornak van egy bizonyos makrónyelve, makró utasítások vannak benne. Ez nem c++ szintaktika.

Az értelmezett # utasításokat végrehajtja, és az eredményt visszaírja a kódba.

Feladatai:

* $\#define$ direktívák kiértékelése
* $\#include$ direktívák hatására megtörténik a fejállományok beillesztése.
* Feltételes direktívák kiértékelése. ($\#if$, $\#ifdef$, stb.)
* Kódformázás (kommentek törlése, indentálás, whitespace sűrítése minimálisra)

## Compiler

Előfeldolgozott forrásfájlt kapja.

Eredménye: $.o$ object file (tárgykód). ASM nyelven, külső hivatkozásokkal (például másik fájlban levő funkciókra mutat hivatkozás). E miatt lesz platform-független a C++. (Linux, Mac, Windows, stb)

Fordítási egység: "Egy forrásfájlba tett változók, függvények halmaza."

## Assembler

Ez csinálja majd a végső gépi kódot. Amilyen gépre fordítjuk, olyan assembler fogja fordítani a célprocesszorokra a programot.

## Linker

Miután minden forrásfájlt $.o$ fájllá alakítottunk, jön a szerkesztés.

Lehet debug vagy release (végleges) verzió.

Windowson ez az $.exe$ eredmény fájl a fordítás után.

A külső hivatkozásokat összekapcsolja.

Libraries: A hivatkozott $.dll$ fájlokkal nem foglalkozunk.

## Mix&match

Meg lehet írni a program egy részét JAVA-ban, másik részét c++-ban.

A kapott object fájlokat össze lehet kapcsolni.

Pl.: F22 célzó rendszerét 80%-át Ada-ban, a maradékot C++-ban írták.

Következő órán nem lesz pluszminusz.