# 1. előadás

Előadó: Pataki Norbert patakino@elte.hu

Időpont: 16:00-17:30

C++ '11 lesz, ha marad rá idő.

## Követelményrendszer

### Fizikusok

Gyakjegy: beadandó (+teszt?)

### Proginfesek

Opcionális beadandó Nem kötelező, de hasznos. Nem jár érte semmi. De leellenőrzik. Mi az, ami nem szép a megoldásában. Ez jobb, mint a funkcionális ("működik") ellenőrzés.

Pluszminusz helyett "pluszplusz". Lehet kapni: -, 0, + (4 db, 10 perc, kódok, elmélet, bármi) Aki mínuszban van félév végén, az a géptermin többet kell, teljesítsen. Elméleti rész minimum pontszáma annyival lesz magasabb, amennyire mínuszban van.

Géptermi ZH: 1 órás beugrós / tesztes rész $15$ kérdéssel, ($4$ válaszból $1$ jót kiválasztani), ebből nyolcat kell eltalálni. (Aki $-4$-re áll, annak $12$-t.) Utána 3 órás gyakorlati feladat. Internet, segédanyag használható! (Beugrón nem.) Kikommentezve meg lesz adva, hogy egy-egy jegyszintért mit kell tudnia a programnak.

Használt fordítóprogram: gcc vagy g++ (command line) kötelező
Nem lehet IDE-ket használni. (CodeBlocks, NetBeans, Microsoft Visual Studio, stb)

Táblás gyakorlat lesz. :(

## Ajánlott irodalom

Bjarne Stroustrup: A C++ Programozási nyelv (1305 oldalas) Angolul jobb.

Scott Meyers: Hatékony C++ (könnyebb)

## Miért vagyunk mi itt?

A C++ zárójelben van a Programozási nyelvek után. Nem a C++ szerepel a középpontban.

A programozási nyelveket szeretnénk megismerni úgy általában, ezt tesszük a C++ szemszögéből.

Több ezer nyelvet terveztek az elmúlt évek alatt. Ebből ma használunk olyan hatvanat.

A jövőben új nyelvek lesznek, ezeket meg kell tudnunk tanulni. Ugyan úgy lesz benne paraméter átadás, OOP, stb.

A C++ elég bonyolult, komplex nyelv ahhoz, hogy csak felejteni kelljen belőle, hogy a mai nyelvekhez eljussunk.

## Tematika

* Programozási paradigmák
* Lexikális egységek
* Paraméterátadási lehetőségek
* OOP
* Típussal való paraméterezés
* Hogyan épül fel egy programozási nyelv (…)
* Öröklődés
* (…)

# Programozási nyelvek / paradigmák története

Paradigmák végiggondolása, egymásra épülése a fontos.

Absztrakciók mentén közelítjük meg a problémát.

## Paradigma

Probléma felbontásának eszköze. Absztrakció.

Egy-egy paradigmához mi tartozik:

* Modellező, támogató eszközök (UML diagram, struktogram, stb.)

Ada Augusta volt az első programozó. '45-re a matematikai alapokat (modellt) akkorra már mind kidolgozták. Ezek közül az első:

## Imperatív programozás

A processzornak van egy adott utasításkészlete, ilyen szintre bontjuk le a programot.

Ilyen a gépi kód, ASM. Kiment a divatból.

Pl.: 0x90 ↔ NOP

## Procedurális programozás

Azt nézzük, a feladat kisebb részekre bontása során milyen függvényekre, alprogramokra lehet bontani a feladatot.

Ilyen a FORTRAN. (És a Pascal, c, stb.)

## Strukturált programozás

Nincs benne $goto$. Feltétel nélküli vezérlésátadást veszélyesnek ítélték. De ma megint kivételeket dobálunk mindenhol. (Exception)

## Objektum Orientált Programozás

Lehetőségeink kibővítése, hogy átláthatóbb, karbantarthatóbb, modulokra felbontható programokat tudjunk írni.

Ilyen a C#, JAVA, C++, stb.

Még mindig gépi kódra fordul vissza.

Típushoz kapcsolódhat művelethalmaz, pl.: $integer$-hez a $.toString()$.

# Más ágak

Minél magasabb az absztrakció, annál lassabb a futó program. Gépi kódban tudjuk a leghatékonyabb programokat megírni.

Nem mindig építünk.

## Logikai programozás

Tételeket, tényeket (clause), igazállításokat definiál a program. Eldönti, hogy igaz-e. (Bebizonyítható-e egy tétel.)

Ilyen a Prolog.

## Funkcionális programozás

Mellékhatás-mentes matematikai függvények (esetszétválasztás, rekurzió) kompozíciójának kiértékelése.

Gráfot épít fel. Lusta kiértékelés nagy előny: Csak akkor számolja ki az "összes páros számot", ha szükség van rá. Haskell-ben a $\left[2,4…\right]$ kifejezés nem ad hibát.

Ilyen a Lisp, Haskell, Erlang.

## Deklaratív programozás

Adatbázis-kezelők nyelve. Csak azt kell definiálnunk, milyen eredményt szeretnénk. Például nem kell megmondani, hogyan kell render​elni a listákat, az a böngésző dolga. Mi csak annyit mondunk: $<ul><li>Elem</li></ul>$.

Ilyen az SQL, HTML.

# Tipikus programozási nyelvek paradigmái

## C++

Multiparadigmás programozási nyelv. Egyszerre több paradigmát is támogat.

* Procedurális
* OOP, de nem kötelező. (Ellenben minden JAVA program kötelezően tartalmaz egy osztályt.)
* Generikus: Adatszerkezeteket és algoritmusokat is ki lehet bővíteni. (Wikipedia szerint: Később is lehet definiálni a funkciókat, fordításhoz elég deklarálni.)
* Metaprogramozás: Nem a programot programozzuk, hanem a fordítót. (Template: Típusparaméterrel lehet ellátni programegységeket. A fordító fogja ezeket a sablonokat feloldani, példányosítani. Ez által kódot generál.)
* Aspektus orientáltság (AOP)

Aspektusszövés: Minden függvény log írással kezdődött, ezt kirakták máshova.

Öröklődés: Új osztály létrehozása könnyű, csak megnézzük, honnan kell leszármaztatni.

Egyéb említett szavak: Osztályhierarchia, design pattern.

Metaprogramozás: x írt egy programot, ami fordulás közben prímszámokat generált és hibaüzenetekben adta ki azokat.

## Érdekességek

1998: C++ első szabványa. 2003-ban lett az a szabvány felülbírálva. 2011-ben jelent meg a C++ '11 párhuzamos programok (parallel computing).

Sikerült megőrizni a c hatékonyságát és "biztonságosságát".

C++-ban a hatékonyság a fontos, a biztonság nem annyira. Inkább a programozó nézzen oda kétszer.

A JAVA biztonságosabb, de cserébe kevésbé hatékony.

Ma már a hardver olcsó. De a mobil eszközök még nem elég gyorsak a "pazarló" nyelvekhez.

### Cloud computing

Drága a "felhőben" futni, megint megtérül a hatékonyság.