C/C++ nyelvi alapok

Konzultáció

2. alkalom

Gyors, gyors, de van gyorsabb..

- Assembly
- https://9gag.com/gag/axVmzgK

C tulajdonságok

- Nincs beépítve
  - I/O kezelés
  - Sztring kezelés
  - Matematikai függvények
- De (!) gazdagon kínál standard függvénykönyvtárakat
- Függvényhívások széleskörű használata
- A típus hanyagolása
- Pointer széleskörű használata

Alapfogalmak I

- Deklaráció:
  - Egy forráskódnak az a részét, amely egy azonosítót egy változóhoz, alprogramhoz, modulhoz, vagy más nyelvi elemhez köt. A deklaráció általában magába foglalja az elem típusának a meghatározását is.
  - Emberül: egy adott objektum első leírása, vagyis amikor definiáljuk azt a valamit. Ekkor nevet is rendelünk ehhez a komponenshez

```
int a;
```

Alapfogalmak II

- Definíció:
  - A definíció a programnyelvekben legtöbbször egy szimbólum (alprogram, művelet, típus) felépítésének, vagy működésének pontos leírását jelenti. A deklaráció párjaként szokták emlegetni, amely csak az adott jel (függvény, típus) megnevezését, felületes leírását tartalmazza.
  - Emberül: egy adott objektumnak amikor értéket adunk, vagyis a komonens azonosítójához értéket rendelünk.

```
a = 4;
```

Alapfogalmak III

- Szintaxis:
  - Formai szabályok olyan rendszerét, amely meghatározza, hogy egy adott kommunikációs nyelvben melyek a szabályos jelsorozatok, a nyelv szintaxisának nevezzük.
  - Emberül: mi az amit leírhhatunk.
- Szemantika:
  - Egy nyelv szemantikája határozza meg, hogy a szintaktikusan helyes jelsorozatok mit jelentenek.
  - Emberül: amit leírtunk az mit jelent.

C felépítés

- Deklarálhatunk:
  - Adattípust

```
typedef unsigned long int ui32;
```

  - Változót

```
int a;
```

  - Függvényt

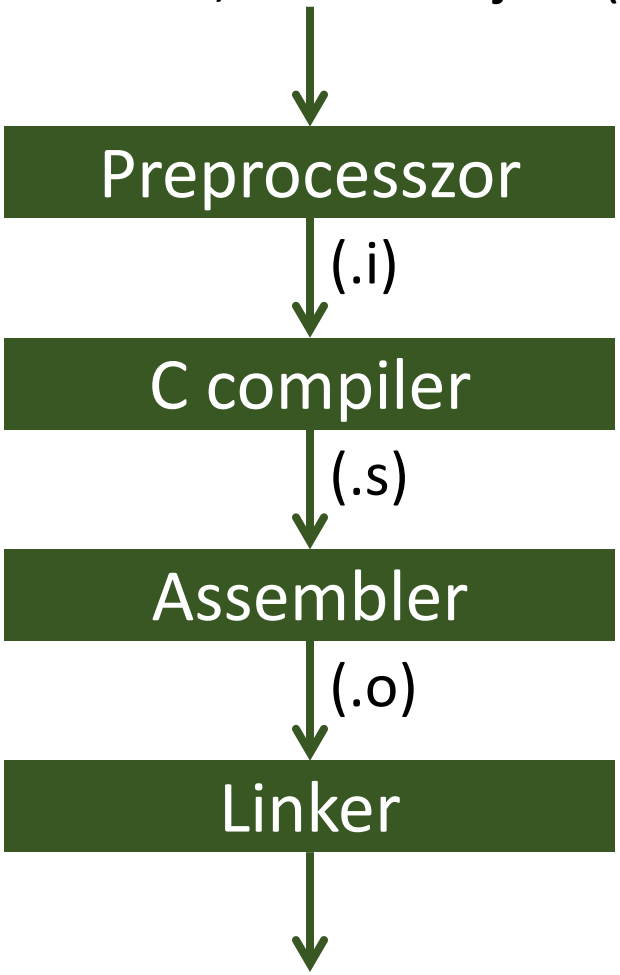
```
void foo(int, double);
```

  - Konstanst

```
#define N 42
```

C felépítés – fordítás (újra)

Forráskód, header fájlok (.c, .h)



C felépítés – header fájlok I

- A C preprocesszor nincs tisztában a C nyelv szintaxisával, csupán egy szövegszerkesztőről van szó. Főbb feladatai:
  - Eltünteti a kommenteket
  - Makro behelyettesítés
  - Feltételes fordítás
  - **Header fájlok bemásolása**

C felépítés – header fájlok II

- A preprocesszornak az **#include** parancs segítségével tudjuk megmondani, hogy a forráskód adott helyére egy másik fájl tartalmát másolja be.
- Header fájlok: konstans- és típus definíciók, függvény- és változó deklarációk szerepelnek, melyek a program fordításához szükségesek
- Két féle header:
  - Szabványos header fájlok

```
#include <stdio.h>
```

  - Saját header fájlok

```
#include "header.h"
```

C felépítés – pointerek

- Egy adott típusú pointer értéke egy olyan memóriaterület címe, ahol a pointer típusa által adott típusú elem(ek) van(nak).
  - A pointer értéke tehát egy dinamikus változó
  - Pointer típusú változót a * segítségével deklarálhatunk:

```
int *p;
```

- Egy változó memóriacíme annak elhelyezkedése a memóriában
  - Ez egy szám, hivatkozni a & jellel lehet rá

```
int a = &p;
```

C ki- és bevitel

- Ki- és bevitel használatához szükségünk van az #include <stdio.h> header fájlra.
- Használhatjuk az alábbi két függvényt (ezeken kívül van még sok más):
  - `scanf(...)`: a bemenetről tudunk olvasni
  - `printf(...)`: a kimenetre tudunk írni
- Mindkét függvény első paramétere egy úgynevezett **formátumsztring**

C ki- és bevitel: formátumsztring

- %d int típusú egész érték
- %f float típusú valós érték
- %lf double típusú valós érték
- %c karakter típusú egész érték
- %s típusú karaktersorozat

C ki- és bevitel: printf()

- Visszatérési értéke: kiírt karakterek száma, hiba esetén negatív érték
- Kiíráskor speciálisan kezelt (vezérlő) karakterek:
  - \n : új sor kezdése
  - \t : ugrás a következő tabulátor pozícióra

```
int a = 4;
printf("a értéke: %d", a);
printf("újsor: \n");
```

C ki- és bevitel: scanf()

- Visszatérési értéke: A beolvasott és sikeresen eltárolt értékek száma, hiba esetén EOF
- A standard bemenetről (stdin) a formátumsztring alapján beolvassa az aktuális paraméterek értékeit és eltárolja azokat a paraméterlistában megadott **memóriacímeken**
- Fontos tehát, hogy beolvasáskor a változó értéke helyett annak **címét** adjuk meg

```
int a;
scanf("%d", &a);
```

C függvény argumentumok I

- Függvény argumentumok lehetnek ki- és bemenőek is.
  - A **bemenő** argumentumokkal adatot adunk át a függvénynek, de ha azokat a függvényen belül módosítjuk, annak nem lesz hatása az eredeti változóra
  - A **kimenő** argumentumon ha bármi változás történik, az az eredeti változóra is kihatással van.

C függvény argumentumok II

- A két mód közötti különbség, hogy a bemenőnél a paramétereket **érték** szerint adjuk át, kimenőnél pedig **referencia** (cím) szerint

```
void csere ( int x, int y)
{
        int m;
        m = x;
        x = y;
        y = m;
}
csere(a, b);

// nem csinál semmit
```

```
csere ( int *x, int *y)
{
        int m;
        m = *x;
        *x = *y;
        *y = m;
}
csere(&a, &b);

// felcseréli a változók
értékét
```