 **A C fordítóprogramos nyelv. Ez mit jelent?**

A fordítóprogramos nyelv (compiled language) azt jelenti, hogy a C forráskódot a fordítóprogram lefordítja gépi kódra (binaris kód), amit az adott operációs rendszer végre tud hajtani. Ez a folyamat előzetesen megtörténik, mielőtt a program futtatásra kerül. A fordított fájlokat általában .exe (Windows) vagy bináris formátumban (Linux/Mac) tárolják, és közvetlenül futtathatók.

 **Interpreteres nyelv. Erre mi jellemző?**

Az interpreteres nyelvben a kódot nem előre fordítják le gépi kódra, hanem egy interpreter (értelmező) futtatja soronként a forráskódot. Ez azt jelenti, hogy a kód végrehajtása közvetlenül történik az interpreter által, amely értelmezi és végrehajtja a kódot menet közben. Például a Python, PHP és JavaScript interpreteres nyelvek.

 **Imperatív nyelv**: Az imperatív programozási nyelvek olyan nyelvek, ahol a programozó lépésről lépésre, utasítások sorozatával határozza meg a számítógép számára, hogy mit kell tennie. Az imperatív programozás során a program végrehajtása egy állapotok sorozatán megy keresztül, ahol az egyes utasítások módosítják a program állapotát. A C nyelv is egy imperatív nyelv.

 **Eljárás-orientált nyelv**: Az eljárás-orientált nyelvek az imperatív nyelvek egyik alfajtája, ahol a programokat eljárásokba, vagy más néven függvényekbe szervezik. Az eljárások különböző feladatokat látnak el, és a program során hívhatók meg. A C nyelv is eljárás-orientált, mivel lehetővé teszi a függvények írását és használatát a kód szervezésére és újrafelhasználására.

 **#include <stdio.h>**:

* Az #include direktíva azt jelenti, hogy a stdio.h (standard input-output library header) fájlt be kell olvasni a programba. Ez a fájl tartalmazza azokat a függvényeket, amelyek lehetővé teszik például a printf függvény használatát, amellyel szöveget írhatunk ki a képernyőre.

 **int**:

* Az int a main függvény visszatérési értékének típusát határozza meg, amely egész szám (integer) típusú. A C nyelv szabványai szerint a main függvénynek mindig vissza kell térnie egy értékkel, általában egy egész számmal, ami jelzi, hogy a program sikeresen lefutott vagy sem.

 **main()**:

* A main a C program belépési pontja, vagyis minden C program ezzel a függvénnyel kezdődik. Ez a függvény hajtódik végre először, amikor a programot elindítjuk. A zárójelek (()) arra utalnak, hogy ez egy függvény, amely paramétereket is fogadhat (de itt nincs paraméter).

 **printf("Hello, World!\n");**:

* A printf függvény egy beépített függvény a stdio.h könyvtárban, amely formázott szöveget ír ki a konzolra. Ebben az esetben a "Hello, World!" üzenetet írja ki, és a \n karakter azt jelenti, hogy a kiírás után egy új sor kezdődik.

 **return 0;**:

* Ez a sor azt jelzi, hogy a main függvény 0-t ad vissza, ami a program sikeres végrehajtását jelzi. A return utasítás mindig megszakítja a függvény végrehajtását, és visszatér egy értékkel. A 0 érték általában azt jelenti, hogy a program hibamentesen futott le. A képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

  Automatikusan generált leírásA képen szöveg, elektronika, képernyőkép, szoftver látható

  Automatikusan generált leírásA képen szöveg, képernyőkép, szoftver, Multimédiás szoftver látható

  Automatikusan generált leírás

int adattípus:

*  **Hány bites?**: Az int adattípus általában 32 bites (4 bájt) a legtöbb modern rendszeren.
*  **Hány byte-os?**: 4 byte (32 bit) az int mérete.
*  **Előjeles – mit jelent?**: Az int előjeles (signed), tehát tárolhat pozitív és negatív értékeket is. Az előjeles tárolás során az egyik bitet (általában a legfelső bitet) az előjel tárolására használják.
*  **Mi a legnagyobb érték, amit tárolni tud?**: Az int adattípus által tárolható maximális érték az előjeles int esetén 2,147,483,647 (INT\_MAX), míg a legkisebb érték -2,147,483,648 (INT\_MIN).
*  **Mutató (p) használata**: A mutató egy olyan változó, amely egy másik változó memóriacímét tárolja. Az ábrán látható, hogy van egy x változó, amelynek kezdeti értéke 5, és egy mutató, p, amely az x változó címére mutat. A mutató segítségével módosíthatjuk az x változó értékét anélkül, hogy közvetlenül hozzáférnénk.
*  **& operátor**: Az & operátor a változó memóriacímét adja vissza. Például, ha van egy x változó, akkor &x az x memóriacímét jelenti, amelyet egy mutatóban tárolhatunk.
*  **\* operátor**: A \* operátor egy mutató által mutatott változó értékét adja vissza. Ez az úgynevezett dereferálás. Ha van egy mutatónk, amely egy változó címére mutat, a \* segítségével elérhetjük a mutatott változó értékét, és akár módosíthatjuk is azt.

**1. Stack és Heap memória-területek:**

* **Stack**: A stack egy korlátozott méretű memória-terület, amelyet a program futásidejű változóihoz használunk, például függvényparaméterekhez, helyi változókhoz. A stack mérete rendszerint kisebb, és automatikusan kezeli a memóriát. A stackből való memóriafoglalás LIFO (Last In, First Out) elven működik.
* **Heap**: A heap egy dinamikus memória-terület, amelyből a program futás közben igényelhet memóriát. A heap használata manuális memóriafoglalást igényel, amelyet például a malloc függvénnyel végzünk, és a memória felszabadítását a free függvénnyel.

**2. malloc és free használata:**

* A malloc (memory allocation) függvény memóriát foglal a heap területéről, a lefoglalt memória méretét byte-ban kell megadni.
* A free függvény a malloc által lefoglalt memóriát szabadítja fel, amikor már nincs rá szükség.