Pointerek a C nyelvben

A pointer a C nyelv egyik kulcsfontosságú fogalma, amely lehetővé teszi a programozók számára, hogy közvetlenül kezeljék/elérjék a memóriát. Más nyelvekben a memória közvetlenül “elérhetetlen” (pl. Java, C#, JavaScript, Python).

A **pointer** *egy memóriacímre mutató változó*, ami lehetővé teszi, hogy közvetlenül hivatkozzunk egy adott memóriaterületre.

Pointer Deklaráció

A pointer deklarációja hasonló a változók deklarációjához, de a típusát is meg kell adni, és a név elé egy **csillagot** írunk.

Például: ha a pointer egy egész szám memóriacímére mutató pointer:

```
int *pointer;
```

Pointer inicializálás

A pointer inicializálása a memóriacímmel történik. Például:

```
int variable = 10;
int *pointer = &variable;
```

Most a pointer változó a 'variable' memóriacímére mutat.

Dereferencia

A 'dereferenciálás' segítségével elérhetjük a pointer által mutatott értéket. Például:

```
int value = *pointer;
```

Most a *value* változó azon értéket tartalmazza, amelyre a *pointer* mutat.

Feladat 1: Definiáljunk egy változót és írassuk ki azt a címet, ahol éppen tárolva van a memóriában.

```
#include <stdio.h>

int main()
{
    int a = 12;
    printf("%p\n", &a);
}
```

Megjegyzés: A **printf(“%p”)** 16-os (hexadecimális) számrendszerben kiírja a címet. Egy változó címét a neve elé írt &-jel jelenti. Ha csak 'a'-t írunk akkor 00000C-fog megjelenni, ami a 12 hexadecimális alakja, azaz a változó értéke, nem a címe lesz.

Feladat 2: Definiáljunk egy változót és egy erre a változóra mutató pointert. Írassuk ki a változó értékét a pointer segítségével.

```
#include <stdio.h>

int main()
{
    int a = 12;
    int *b; // egy egészre mutató pointer létrehozása
    b = &a; // a pointer mutasson az 'a' változó címére
    printf("%d\n", *b);  // a b előtti * mutatja, hogy nem a címet, hanem az
ott tárolt értéket kell kiíratni
}
```

Feladat 3: Az előző feladatban adott b pointer segítségével növeljük meg az **a** változó értékét, majd írassuk ki az **a** változót.

```
#include <stdio.h>

int main()
{
    int a = 12;
    int *b; // egy egészre mutató pointer létrehozása
    b = &a; // a pointer mutasson az 'a' változó címére

    *b = *b + 1;

    printf("%d\n", a);
}
```

Érték növelése pointeren kereszül

```
#include <stdio.h>

int main()
{
    int a = 10;
    int* b = &a;

    // növeljük meg a értékét b-n keresztül röviden írva
```

```
    *b++;

    printf("%d", a);
}
```

Nem lett 11 **a** értéke. Miért? Mert (*b)++ -t kellett volna írni.

Érdekesség: Pointerre mutató pointer

Legyen egy **val** változó értéke 10. Hozzunk létre egy pointert ami a val változó címére mutat. Hozzunk létre egy másik pointert ami az első pointerre mutat. Irjuk ki a pointerek által mutatott címen tárolt értéket!

```
#include <stdio.h>

int main ()
{
    int  val = 10;
    int  *ptr;
    int  **pptr;

    ptr = &val;
    pptr = &ptr;

    printf("val erteke = %d\n", val );
    printf("*ptr = %d\n", *ptr );
    printf("**pptr = %d\n", **pptr);
}
```

Érdekesség: Jelszó feltörés

Az alábbi program bekér egy karaktert a felhasználótól. Ha nem c-t nyom akkor nem helyes a kódja.

```
#include <stdio.h>

int main()
{
    char kod;
    int helyes = 0;

    printf("Kerem a varazsbillentyu megnyomasat.\n Csak 1 probalkozasa van,
Mr. Bond.\n");
    scanf("%s", &kod);

    if( kod == 'c' )
    {
        printf ("A jelszo helyes\n");
```

```
        helyes = 1;
    }
    else
    {
        printf ("A jelszo helytelen\n");
    }

    if(helyes)
    {
        printf ("On a rendszergazda\n");
    }
}
```

Látva a kódot, ha tegyük fel 5db bármilyen karaktert ütünk be, akkor is rendszergazdák leszünk. (érdemes kikísérletezni hány darab leütés kell, hogy elromoljon a működés) Magyarázzuk meg miért? (azért, mert a scanf felülírja a memóriát és a “helyes” érték 0-ja felülíródik. Ezt nevezzük “buffer túlcsordulásos” támadásnak.

From:
https://edu.iit.uni-miskolc.hu/ - **Institute of Information Science - University of Miskolc**

Permanent link:
https://edu.iit.uni-miskolc.hu/tanszek:oktatas:szamitastechnika:mutatok_pointerek

Last update: **2023/11/14 11:10**