A változók pontos megadása

A következő érdekes példák a változók pontos típusmegadására figyelmeztetnek.

Feladat: Változók definíciója és alapvető számolások

```
#include <stdio.h>

int main ()
{
    int a, b;
    int c;
    float f;

    a = 10;
    b = 20;

    c = a + b;
    printf("c = %d \n", c);

    f = 70.0 / 3.0;
    printf("f = %f \n", f);
}
```

Megjegyzés: Jegyezzük meg, hogy a 10 általában minden programozási nyelvben egész számot jelent, binárisan 32 biten kódolva:

00000000 00000000 00000000 00001010 lesz.

A 10.0 jelenti a double típusú lebegőpontos számot ami 64 bites és így nézne ki:

0 10000000010 0100000000000000000000000000000000000000000000000000

és a 10.0f (igen, a végére egy f betűt írunk), jelenti a *float* típusú lebegőpontos számot ami ugyancsak 32 bites, de így néz ki binárisan:

0 10000010 01000000000000000000

A szóközök az előjel az exponens és a mantissza tagokat választják el egymástól. Látható tehát hogy a 10 10.0 10.0f három teljesen különböző módon van értelmezve kettes számrendszerben.

f = 70.0 / 3.0; helyett ha ezt írjuk: **f = 70 / 3;** ezért fog más eredményt adni hiszen így elvesztjük a tizedesjegyeket.

1. Feladat: Hibásan számol a c nyelv??? Mi lehet a hiba?

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

```
int main ()
{
    int a = 13;
    int b = 3;
    int c;
    c = a / b;
    printf("c = %d \n", c);
}
```

A 3 nem osztója a 13-nak. A 13 / 3 eredményét a c változóban egészként tároljuk, ilyenkor az eredmény egész részét aza a 4-et fogjuk eredményként kapni.

Ahhoz hogy a tizedes jegyeket is megkapjuk, próbáljunk meg áttérni float adattípusra, a következő példa alapján.

2. Feladat: Az előző kódban változtassuk meg a c-t float-ra és a printf()-ben használjunk %f-et és próbáljuk ki úgy is. Segített?

```
#include <stdio.h>

int main ()
{
    int a = 13;
    int b = 3;
    float c;
    c = a / b;
    printf("c = %f \n", c);
}
```

Azért nem segít, mert az a / b elvégzésekor, mivel mind a két érték egész, az eredmény megint 4 lesz. A c-t hiába adtuk meg float-ként, akkor is a 4-es kerül bele.

3. Feladat: Hozzon létre egy **unsigned int** változót 2 illetve egy **int** változót -1 értékkel és végezzen összehasonlítást, hogy 2 > -1? Magyarázza meg, hogy mi történik és miért?

```
#include <stdio.h>

int main()
{
    int a = -1;
    unsigned int b = 2;
    if ( b > a)
    {
       printf("a 2 nagyobb, mint a -1");
    }
    else
```

```
    {
       printf("a 2 kisebb, mint a -1 ???");
    }
}
```

4. Feladat: Miért számol hibásan (pontatlanul) a következő kód?

```
#include <stdio.h>

int main()
{
    float a = 4.20;
    float b = 0.25;
    printf("%f\n", a / b);
}
```

Nézzük meg hogyan ábrázoljuk a változókat: https://www.h-schmidt.net/FloatConverter/IEEE754.html

Írjuk be a mezőbe a két értéket és figyeljük meg a hibát.

Hogyan lehetne átalakítani a forráskódot, hogy pontosan 16.800000 legyen az eredmény?

Magyarázzuk meg miért nem működik a (double) (a/b) típuskonverzió?

5. Feladat: Miért számol hibásan (pontatlanul) a következő kód?

```
#include <stdio.h>
#define N 3 + 5
int main()
{
    int a = 2;

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

Az eredménynek 2 * 8 nak kellene lennie, de 11-et ír ki, mi lehet a hiba? Az a probléma, hogy a #define nem végez műveletet, hanem egyszerűen behelyettesít, így az a * N művelet a * 3 + 5 lesz, ami a *precedencia* (sorrendi) szabályok miatt 6 + 5 = 11 lesz.

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:egyszeru_mintapeldak

Last update: **2026/10/14 06:59**