```c
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
int main()
{
    system("chcp 1250 > nul");
    setlocale(LC_ALL, "");

    //kiírások
    printf("Variables of type char occupy %u bytes\n", sizeof(char));
    printf("Variables of type short occupy %u bytes\n", sizeof(short));
    printf("Variables of type int occupy %u bytes\n", sizeof(int));
    printf("Variables of type long occupy %u bytes\n", sizeof(long));
    printf("Variables of type long long occupy %u bytes\n", sizeof(long long));
    printf("Variables of type float occupy %u bytes\n", sizeof(float));
    printf("Variables of type double occupy %u bytes\n", sizeof(double));
    printf("Variables of type long double occupy %u bytes\n", sizeof(long
double));

    return 0;
}
```

A program kiírja az egyes beépített C típusok (pl. `int`, `char`, `double` stb.) által elfoglalt memóriaméretet **bájtban** a `sizeof()` operátorral.

Miért jó és hasznos így?

1. Átlátható típusméret-vizsgálat

- Segít megérteni, hogy **melyik típus mennyi memóriát foglal**, ami fontos:
  - hatékony memóriakezelésnél,
  - struktúrák, fájlformátumok, bináris adatok kezelésénél.

2. Platformfüggetlen tanuláshoz tökéletes

- A `sizeof()` **a futtatott rendszer szerint** adja vissza a méretet.
  - Pl.: `int` lehet 4 bájt 32 és 64 biten is, de `long` változhat: 4 vagy 8 bájt.

3. Automatikus és pontos

- Nem kell kitalálni vagy feltételezni a típusról semmit.
- Fordítótól és platformtól függetlenül **pontos információt kapsz**.

4. Tanuláshoz és hibakereséshez szuper

- Ha nem érted, miért nem fér bele egy szám egy típusba → ezzel ellenőrizheted.

Kis javaslat a printf-hez

Használj `%zu` helyett `%u` **helyett**, mert a `sizeof()` visszatérési típusa:
👉 `size_t` (ami `unsigned` típus, gyakran `unsigned long` vagy `unsigned long long`)

Tehát így még szebb és biztonságosabb:

```
printf("Variables of type int occupy %zu bytes\n", sizeof(int));
```

De működik `%u`-val is **Windows alatt**, mert ott `size_t` gyakran `unsigned int`.

Alternatíva: típussal és mérettel együtt sorban

```
printf("%-20s : %2zu bytes\n", "int", sizeof(int));
```

Így táblázatszerűen láthatod a különbségeket, pl.:

```
int                  : 4 bytes
long                 : 4 bytes
long long            : 8 bytes
```
