Kis kitekintés: a standard könyvtár

Röviden megnézzük azt a két függvényt, ami szükséges a továbbiakban, hiszen nem csak számolásokat, algoritmusokat akarunk implentálni, hanem a felhasználóval (beleértve saját magunkat is) kommunikáló interaktív alkalmazásokat is. Ezeket természetesen saját magunk is implementálhatnánk (kicsit nagyobb ismerettel a jelenleginél).

Előfeltétel: Legyen a helloworld programunk megírva.

Rövidítés: A továbbiakban a fordítsuk le a programot és futtassuk a generált futtatható állomámyt a következő rövidítéssel értjük: ***C&R*** (compile and run).

printf

A helloworld-ben már használtuk, a standard ouptutra dolgozó függvény. Deklarációja a következő:

*int printf (const char * format, ...);*

A visszatérési értéke a kiírt karakterek száma, ellenőrzésekhez remekül használható. A nem szokványos argumentumlistája a variadic függvényekre jellemző, azaz tetszőleges számú argumentumot is adhatunk neki. A helloworld alkalmával csak egy argumentumot, a *format* azonosítójút adtuk át. Ez a kiírandó szöveg kinézetét határozza meg, valamint magát a szöveget, amit ki szeretnénk iratni.

A *format* argumentumban további speciális karaktereket is megadhatunk, melyeket a *%* karakterrel jelzünk, folytatva előre meghatározott karakterrel (jelöljük ezt most %_, ahol a _ karakter helyettesítve lesz az előre meghatározott karakterrel). Minden *%_* után a *printf* vár egy új argumentumot, melynek típusa (lásd változóknál) a _ karakterrel van meghatározva.
A fontosabb esetek:

- *%d :* egész típust vár az argumentumnak

  Feladat: módosítsuk a helloworld.c-ben a *printf* függvényt, majd *C&R*:

  printf("Hello World in %d!\n", 2021);

- %*c* : karakter típust vár az argumentumnak

  Feladat: módosítsuk a helloworld.c-ben a *printf* függvényt, majd *C&R*:

  printf("%cello World!\n", 'H');

  Megjegyzés: A C-ben a ' ' között karaktereket, " " között karakterláncokat adunk meg.

- %s : karakterláncot vár az argumentumnak

  Feladat: módosítsuk a helloworld.c-ben a *printf* függvényt, majd *C&R*:

  printf("Hello %s", "World!\n");

- %f : lebegőpontos típust vár az argumentumnak

  Feladat: módosítsuk a helloworld.c-ben a *printf* függvényt, majd *C&R*:

  printf("Hello %f part of World!\n", 0.001);

A helloworld-ben látott újsor karakter nem tekinthető speciálisnak, mivel nem növelik az argumentumlistát. Egyszerűen ez a karakter máshogy nem íratható ki a *printf* függvénnyel. Hasonlóan több más karakter se, ilyenek a -teljesség igénye nékül- a \t tabulátor, a \\ backslash karakter, a \r carriage return karakter (érdekesség: különböző operációs rendszerekben más és más a mondat végi újsor karakter, DOS alapú rendszereknél (pl Windows) ez a \r\n két karakter kombinációja, Unix alapú (mint a Linux is) csak \n karakter található).
Bővebben: https://en.wikipedia.org/wiki/Escape_sequences_in_C

Mint láttuk, a *printf* argumentumlistája tetszőleges hosszú lehet. Ebből könnyű kitalálni, hogy tetszőleges számosságú speciális %_ is megadható a format argumentumon belül.

Feladatok:

- módosítsuk a helloworld.c-ben a *printf* függvényt, majd *C&R*:

  printf("Hello %s in %d!%c", "World", 2021, '\n');

- módosítsuk a helloworld.c-ben a *printf* függvényt, majd *C&R*:

  printf("Hello %s in %d", "World!%c", 2021, '\n');

  Megjegyzés: A fenti sorban látható, hogy nem lehet a format argumentumot bővíteni későbbi karakterláncokban elhelyezett speciális %_ karakterekkel.

A *printf* függvény teljes leírása: https://en.cppreference.com/w/c/io/fprintf

scanf

A standard inputról (terminal) bekér adatot a felhasználótól. Deklarációja:

*int scanf (const char * format, ...);*

Szintén variadic függvény, a visszatérési értéke a beolvasott karakterek számával egyezik meg (ha a beolvasás sikeres), ellenőrzésekhez tudjuk használni. A *format* argumentum a *printf* függvénynél bemutatott speciális %_ karaktereket tartalmazhatja, amikkel a beolvasott adat típusát határozhatjuk meg. A további argumentumok pedig a változókat takarja, ahová szeretnénk a beolvasott adatokat letárolni. Valójában a *scanf* számára nem a változót, hanem annak memóriacímét adjuk át, azaz azt a mutatót, amely a változó memóriában elfoglalt helyére mutat (bővebben a *pointerek* tárgyalásánál). Egyelőre fogadjuk el, hogy ezt a & karakterrel és a változó nevével jelöljük a scanf argumentumlistájában.
Valamint előlegezzük meg a változók és típusok létezését is.

Feladat:

- Nyissuk meg a helloworld.c kódunkat és írjuk be az alábbi sorokat, majd *C&R:*

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

  Futáskor a prompt egy egész számot vár értékül, *enter* nyomása után adjuk át az adatot a programnak.
  Irassuk ki ezt a számot a printf függvénnyel:

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

  Próbáljunk más, nem egész számot megadni további futtatásokkal.
  Megjegyzés: Mivel az *a* típusa integer, ha valami más értéket adunk meg, akkor implicit típuskonverzió történik, azaz a program egész számmá alakítja a beadott értéket.

- Módosítsuk az *a* változó típusát float-ra, de ne változtassunk meg mást, majd *C&R*:

```
float a;
```

  A fordító figyelmeztet a *scanf* és *printf* függvénynek átadott *float* típusról, de sikeresen lefordul.
  Megjegyzés: A kiírt érték nem egyezik meg a beírt számmal, ugyanis a lebegőpontos és egész típusok máshogy tárolódnak a memóriában.
  Módosítsuk a kiíró függvényünket is, majd *C&R*:

```
printf("a = %f\n", a);
```

  Továbbra is figyelmeztet a fordító, de már csak a *scanf* függvényre. Az *a* változó kiírt értéke továbbra se egyezik meg a beírt számmal.
  Végül az utolsó javítás, *C&R*:

```
scanf("%f", &a);
```

- Természetesen karakterláncot is megadhatunk a *scanf* függvénynek, megelőlegezve a tömböket, valamint a tömbök és mutatók közti kapcsolatot, ugyanis ekkor nem szükséges a *&* karakter a *scanf*-ben:

```
char a[15];
scanf("%s", a);
printf("a = %s\n", a);
```

  C&R, majd próbáljunk ki több lehetőséget is, egész és lebegőpontos számot megadni, karakterláncot. Vegyük észre, hogy a *white space* karakterlánc határoló, azaz a Hello World bemeneti értéknél csak a Hello karakterláncot írja ki a printf, míg a HelloWorld esetén a teljes karakterlánc kiírásra kerül. Próbáljunk ki speciális karaktereket is (pl. \ vagy %).

A *scanf* függvény teljes leírása: https://en.cppreference.com/w/c/io/fscanf