Instrukcje wejścia/wyjścia

Praktyczny program powinien mieć możliwość interaktywnej komunikacji z użytkownikiem. Do drukowania informacji dla użytkownika służy najczęściej standardowe wyjście (monitor). W nowym projekcie pakietu MS Visual Studio (poproś prowadzącego, aby pokazał, jak stworzyć **pusty** projekt), napisz program, który wydrukuje tekst *Witaj na trzecim laboratorium!*

```
void main()
{
   printf("Witaj na trzecim laboratorium!");
}
```

Instrukcja `printf` służy do wypisywania tekstu na ekran. Jako argument przyjmuje zmienną typu tekstowego. Do formatowania tekstu służą *sekwencje formatujące*, które pozwalają wprowadzić znak nowej linii, tabulacji itp. Umieszczona wewnątrz tekstu sekwencja znaków:

- `"\n"` — wprowadza znak nowej linii,
- `"\t"` — wprowadzan znak tabulacji.

Ćwiczenia

Używając **jednej** instrukcji `printf` oraz odpowiednich sekwencji formatujących, wygeneruj tekst identyczny z poniższym:

```
To jest pierwsze zdanie w mojej instrukcji.
To jest tuz po znaku nowej linii.      Zas ten fragment
                                       oddzielony jest znakiem
                                       tabulacji!
Za to w ponizszej linii wszystkie liczby oddzielono tabulatorami.
5.2    3.14    -7    8
```

Uwaga

Oczywiście wprowadzenie długiego tekstu (np. kilku komunikatów dla użytkownika) w jednej instrukcji `printf` jest nonsensem. Spróbuj osiągnąć ten sam efekt, co powyżej, ale tym razem użyj osobnej instrukcji `printf` dla każdego ze zdań. Czy coś cię zaskakuje? Czy nowa instrukcja `printf` wymusza przejście do nowej linii?

W instrukcji `printf` nie używaj polskich znaków diakrytycznych. Da się to zrobić, jednak wymaga pewnych komplikacji i w prostych programach nie jest praktykowane. Jeśli bardzo cię męczy ciekawość, w wolnej chwili poszukaj rozwiązań w książkach, bądź internecie.

Dalej o `printf`

Pewne znaki specjalne są w języku C zarezerwowane na potrzeby konkretnych instrukcji. Wiele z nich poznasz wkrótce. Dobrymi przykładami takich znaków są % czy backslash . Nie mogą one być użyte wprost, gdyż mają swoje funkcje w języku C. Jeśli chcesz, by się pojawiły na ekranie, musisz poprzedzić je dodatkowym znakiem . - Dopisz do swojego programu intstrukcję, która wydrukuje następujący tekst:

```
82% dysku C:\ jest w uzyciu!
```

Program o znaczeniu inżynierskim musi jednak mieć możliwość drukowania na ekran liczb i wyników przeprowadzonych działań. - Przepisz do funkcji main następujące instrukcje:

```
int a = 5;
double c = 8.2;

printf("Zmienna a ma wartosc %d, zas zmienna c = %lf\n", a, c);

c = c + 7.5;
c -= a;
a = 1;
c -= 2*a;

printf("Po dodaniu do zmienej c wartosci 7.5, odjeciu a
        oraz odjeciu dwukrotnosci zmodyfikowanej
        wartosci a zmienna c = %lf\n", c);
```

- Przeanalizuj dokładnie kod. Pojawiają się w nim nowe instrukcje arytmetyczne!

- Między wszystkimi instrukcjami arytmetycznymi dodaj po jednej linijce kodu, który wydrukuje na ekran bieżącą wartość przechowywaną w zmiennych a i c. Pojawiły się też nowe elementy. Do drukowania wartości przechowywanych w zmiennych służą *sekwencje formatujące* lub inaczej *specyfikatory formatu*. Są one następujące:
- %lf — dla zmiennych typu `double`
- %d — dla zmiennych typu `int`
- %f — dla zmiennych typu `float` Dodatkowo, dla liczb zmiennoprzecinkowych o ekstremalnie małych, umiarkowanych i ogromnych wartościach użyj poniższych sekwencji i zobacz, jaki będzie efekt działania.
- %lg, %e, %.2lf, %.4lf (dla zmiennych typu `double`),
- %.3f (dla zmiennych typu `float`).

Czytanie z klawiatury

Instrukcją służącą do czytania danych ze standardowego wejścia (klawiatury) jest instrukcja `scanf`. Przykłady jej użycia wyglądają następująco:

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

double c;
scanf("%lf", &c);

int b, d;
double g, h;
scanf("%lf%d%d%lf", &g, &d, &b, &h);
```

Uwaga: Zwróć szczególną uwagę na znak & występujący przed nazwami zmiennych, do których wczytujemy wartości. Znak ten **nigdy** nie występuje w intrukcji `printf`, za to zawsze jest potrzebny w instrukcji `scanf`.\ Zauważ również, że używając jednej instrukcji `scanf` możesz wczytać wiele liczb. Sekwencje formatujące nie muszą być oddzielone spacjami, za to wartości muszą być podane z klawiatury w odpowiedniej kolejności - takiej, w jakiej zmienne na liście argumentów, do których te wartości mają trafić.

Ćwiczenia

Napisz prosty kalkulator, który wczyta z klawiatury dwie liczby typu rzeczywistego i wykona na nich dodawanie, odejmowanie, mnożenie i dzielenie. Odejmowanie i dzielenie oczywiście nie jest przemienne. Policz zatem każdą z możliwych różnic czy ilorazów. Wydrukuj wszystkie wyniki na ekran.

Jeszcze trochę o funkcjach

Funkcje nie tylko grupują pewne logicznie wydzielone bloki instrukcji, których używamy wielokrotnie (jak funkcja rysująca ludzika z kółek i kresek, bądź funkcja rysująca tłum z użyciem funkcji `ludzik`). Do tej pory ich deklaracje i definicje wyglądały odpowiednio tak:

```
void NazwaFunkcji(int argument1, double argument2);

void NazwaFunkcji(int argument1, double argument2)
{
    // Tu sie znajduje cialo funkcji
}
```

Funkcje mogą bowiem zwracać wartość. Typ zmiennej, jaką zwracają jest zawsze identyczny z typem funkcji. Nie musi za to być zgodny z typami argumentów, których typy mogą być zupełnie inne. Weźmy dla przykładu funkcję, która przyjmie dwie wartości (jedną typu `double`, drugą typu `float`) i zwróci liczbę całkowitą równą 5, gdy większą wartość ma pierwszy argument lub wartość 10 w przeciwnym razie. Przeanalizujmy odpowiednio deklarację i kod takiej funkcji.

```
int KtoryWiekszy(double a, float b);

int KtoryWiekszy(double a, float b)
{
   int Wynik;

   if(a > b)
   {
      Wynik = 5;
   }
```

```
   else
   {
      Wynik = 10;
   }
   return Wynik;
}
```

Zwróć uwagę na instrukcję `return`, która zwraca z funkcji **wartość przechowywaną w konkretnej zmiennej**. To ważne! Funkcja nigdy nie zwraca zmiennej. Zwraca tylko wartość, jaka była w tej zmiennej przechowywana. Ponadto zmienna zadeklarowana w danej funkcji będzie dla programu widoczna **tylko i wyłącznie wewnątrz tej funkcji**, a nie będzie rozpoznawana w innych fragmentach kodu (np. funkcji `main`). Prześledźmy jeszcze kod funkcji `main`, w której występuje wywołanie naszej funkcji.

```
void main()
{
   float c = 8.14;
   double d = -7.3814;
   int InnaZmienna = 15;

   KtoryWiekszy(d, c);
   InnaZmienna = KtoryWiekszy(d, c);
   InnaZmienna = KtoryWiekszy(12.5, c);
}
```

Dodaj do powyższego kodu instrukcje, które po każdym wywołaniu funkcji `KtoryWiekszy` wydrukują wartość aktualnie przechowywaną w zmiennej `InnaZmienna`. Zastanów się, jaki będzie wynik i sprawdź, czy masz rację.

Zmodyfikuj napisany dziś kalkulator tak, aby instrukcje sumowania, odejmowania, mnożenia i dzielenia były realizowane przez osobne funkcje `Sumuj`, `Odejmij`, `Pomnoz`, `Podziel`. Funkcje te musisz napisać samodzielnie.

Coś na deser *

Drukowanie tekstów na ekran nie musi sprowadzać się tylko do drukowania napisów, które są na twardo zdefiniowane w kodzie źródłowym lub wartości przechowywanych w zmiennych liczbowych. Język C ma również odpowiedni typ na przechowywanie zmiennych tekstowych, których zawartość może dynamicznie się zmieniać w trakcie wykonywania programu. Spróbuj zrozumieć i skompilować poniższy kod. Więcej szczegółów stanie się dla Ciebie jasnych, gdy omówione zostaną *tablice*.

```
void main()
{
   char tekst[] = "To jest moj tekst\n";

   printf(tekst);
}
```

Istnieje również szereg funkcji, które pozwalają łączyć teksty, porównywać je ze sobą, przekształcać zmienne liczbowe do postaci zmiennych tekstowych. Zainteresowanych odsyłamy do zewnętrznych materiałów poświęconych *zmiennym łańcuchowym (ang. string)*.