Niniejsza instrukcja poświęcona jest trzem zagadnieniom:

- Obsłudze standardowego wejścia i wyjścia,
- Wprowadzeniu do wskaźników,
- Zwracaniu wartości przez funkcje.

Instrukcje wejścia/wyjścia

Praktyczny program powinien mieć możliwość interaktywnej komunikacji z użytkownikiem. W celu drukowania informacji często wykorzystywane jest standardowe wyjście (ekran).

Drukowanie tekstu

Utwórz nowy projekt pakietu MS Visual Studio i napisz program, który wydrukuje tekst *Witaj na trzecim laboratorium!*

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

Funkcja `printf()` służy do wypisywania tekstu na ekran. Jako pierwszy argument przyjmuje sformatowany tekst. Do formatowania tekstu służą *kody sterujące*, które pozwalają wprowadzić znak nowej linii, tabulacji itp. Przykładowo, umieszczona wewnątrz tekstu sekwencja znaków:

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

Ćwiczenia

Używając **pojedynczej** funkcji `printf()` oraz odpowiednich kodów sterujących, 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!
W ponizszej linii wszystkie liczby oddzielono tabulatorami:
5.2    3.14    -7    8
```

Uwaga: jeśli instrukcja jest długa, a przez to mało czytelna, warto kontynuować ją w nowej linii. W tym celu należy posłużyć się ukośnikiem wstecznym „\”. Przykładowo:

```
printf("Ta instrukcja kontynuowana jest \
w kolejnej linii\n);
```

Spróbuj osiągnąć ten sam efekt, co powyżej, ale tym razem użyj osobnej instrukcji `printf` dla każdego ze zdań.

W instrukcji `printf()` nie używaj polskich znaków diakrytycznych. Można to zrobić, jednak ze względu na przenośność w prostych programach nie jest praktykowane.

Znaki specjalne

Pewne znaki są w języku C zarezerwowane na potrzeby wykonania konkretnych operacji. Jeśli chcemy je wydrukować na ekran, nie mogą być użyte wprost. Przykładowo, jeśli chcemy za pomocą funkcji `printf()` wydrukować znaki „%”i „\”należy je zapisać podwójnie, tzn. „%%”i „\\”.

Dopisz do swojego programu instrukcję, która wydrukuje następujący tekst:

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

Drukowane liczb

Przepisz do funkcji main następujące instrukcje:

```
int a = 5;
double b = 8.2, c = 7.5;
```

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

c += b;
c -= a;
a = 1;

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

Zauważ, że do drukowania wartości przechowywanych w zmiennych służą tzw. *specyfikatory formatu*. Na zajęciach najczęściej będą używane:

```
- %lf -- dla zmiennych typu `double`
- %d -- dla zmiennych typu `int`
- %f -- dla zmiennych typu `float`
- %c -- dla zmiennych typu `char`
```

Ćwiczenia

- Dodaj po linijce kodu, który wydrukuje na ekran bieżącą wartość przechowywaną w zmiennych. Co oznaczają operatory `+=` i `-=`?
- 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 funkcja `scanf()`. Przykład jej użycia wygląda następująco:

```
int a;
scanf("%d", &a);
printf("Wczytana liczba: %d\n", a);

double c;
scanf("%lf", &c);
printf("Wczytana liczba: %lf\n", c);

int b, d;
double g, h;
scanf("%lf %d %d %lf", &g, &d, &b, &h);
printf("Wczytane liczby: %lf %d %d %lf\n", g, d, b, h);
```

Uwaga:

- Zwróć szczególną uwagę na znak „&'‘występujący przed nazwami zmiennych, do których wczytujemy wartości. Znak ten służy do ustalenia zmiennej występującej za raz po nim. Funkcja `scanf()` wczytuje dane z klawiatury, które musi gdzieś zapisać. Stosując znak „&” dostarczamy informacji o tym gdzie znajduje się zmienna, do której należy zapisać wczytaną wartość.
- Zauważ również, że używając pojedynczej instrukcji `scanf` możesz wczytać wiele liczb. Należy przy tym pamiętać o podaniu specyfikatorów formatu w takiej kolejności jak zmienne na liście argumentów. Powinno się unikać stosowania innych symboli niż spacje i specyfikatory formatu. Inaczej należy pamiętać o podaniu liczb razem z tymi dodatkowymi symbolami.

Ćwiczenia

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

Wskaźniki

Każdy element programu musi być zapisany gdzieś w pamięci komputera. Oznacza to, że musi mieć swój adres. Adres ten można zapisać w pewnej zmiennej i posłużyć się nim w programie. Zmienną zawierającą adres innej zmiennej nazywamy **wskaźnikiem**. Wskaźniki mogą zawierać adresy różnych elementów (zmiennych, tablic, funkcji, ...). Z tego powodu będą pojawiały się na różnych etapach tego kursu. W tej instrukcji opiszemy jedynie podstawy.

Jak już wspomnieliśmy, każda zmienna ma swój adres. Można go uzyskać stosując operator & i zapisać w zmiennej typu wskaźnikowego. Mówiliśmy o tym podczas

zajmowania się funkcją `scanf()`. Zmienne różnych typów wymagają różnej wielkości pamięci. Dlatego deklaracja wskaźnika zawierającego adres zmiennej danego typu, ma postać `typ *wskaznik;`.

Przykładowo:

```
double x;               // Deklaracja zmiennej typu double
x = 4.2;                // Definicja - nadanie zmiennej wartosci

double *wsk_x;          // Deklaracja wskaznika mogacego przechowac
                        // adres zmiennej typu double
wsk_x = &x;             // Definicja - nadanie zmiennej wartosci
                        // bedacej adresem zmiennej x
printf("%p", wsk_x);    // wydrukowanie adresu przechowywanego przez
                        // wskaznik na ekran
printf("%lf", *wsk_x); // wydrukowanie wartosci zmiennej, ktorej
                        // adres przechowuje wskaznik
printf("%lf", x);       // wydrukowanie wartosci zmiennej x
```

Podsumowując:

- `&x` – pobiera adres zmiennej,
- `*wsk_k`– pobiera wartość zmiennej, na którą wskazuje wskaźnik,
- `"%p"` – informuje funkcję `printf()`, że wyświetlana wartość to adres.

Ćwiczenia

- Stwórz trzy zmienne `a`, `b` i `c` tego samego typu.
- Do zmiennej `a` wczytaj liczbę.
- Utwórz dwa wskaźniki na pozostałe zmienne.
- Wykorzystaj wskaźniki do *przepisania* wartości wczytanej do zmiennej `a` do pozostałych zmiennych.
- Dla sprawdzenia, wyświetl zawartość wszystkich zmiennych.

Funkcje i zwracanie argumentu

Do tej pory, funkcje deklarowaliśmy i definiowaliśmy w tym samym samym miejscu (przed funkcją `main()`). W przypadku gdy kod programu jest długi lub ma być wykorzystany przez innego użytkownika rozdziela się deklarację od definicji. Instrukcja:

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

jest deklaracją (zauważ, że kończy się średnikiem).

Instrukcja:

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

jest definicją i może być umieszczona także za funkcją `main()`.

Funkcje nie tylko grupują pewne logicznie wydzielone bloki instrukcji, których używamy wielokrotnie (jak funkcja rysująca ludzika z kółek i kresek. W takim przypadku wystarczyło zadeklarować, że funkcja nic nie zwraca wykorzystując słowo kluczowe `void`.

Funkcje mogą jednak zwracać także wartość. Typ zmiennej, jaka jest zwracana znajduje się przed nazwą funkcji.

Weźmy dla przykładu funkcję, która przyjmuje dwie wartości (jedną typu `double`, drugą typu `float`) i zwraca 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 służy do zwracania **wartości przechowywanej w konkretnej zmiennej**.

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. wewnątrz 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);
}
```

Ćwiczenia

- Dodaj do powyższego kodu instrukcje, które po każdym wywołaniu funkcji `KtoryWiekszy` wydrukują wartość aktualnie przechowywaną przez zmienną `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` i `Podziel`.