Zmienne

Działanie każdego programu polega na przetwarzaniu danych.
Dane w programie przedstawiamy za pomocą literałów i zmiennych oraz stałych.

Literał - ***to napis w programie reprezentujący w sposób bezpośredni wartość danej***

Na przykład napis:
`111 //literał liczbowy`
reprezentuje liczbę `111` i jest traktowany jako liczba `111`.

Do przechowywania danych w programie i ponownego ich użycia w kolejnych operacjach służą zmienne.

Zmienna ***jest symbolem w programie, oznaczającym obszar w pamięci komputera, w którym mogą być zapisywane różne dane.***

Czyli:

- zmienna ma nazwę (użyty w programie symbol nazywa się nazwą zmiennej),
- przez tę nazwę odwołujemy się do konkretnego obszaru pamięci, w którym chcemy przechowywać wartości jakiejś danej,
- zawartość tego obszaru (wartość zmiennej) możemy zmieniać w trakcie wykonania programu

Deklaracja zmiennych.

Aby stworzyć zmienną należy najpierw zadeklarować, jakiego będzie typu, czyli zdecydować czy ma przechowywać liczby naturalne, całkowite, wymierne itp. lub znaki. Zmienną deklarujemy w następujący sposób:

nazwa_typu nazwa_zmiennej;

np.
```
int x;
```

Jeśli chcemy zadeklarować kilka zmiennych tego samego typu deklarujemy je oddzielając przecinkami:

```
int a, b, c;
```

Słowo kluczowe `int` oznacza, że tworzymy zmienną typu integer (w skrócie `int`), która będzie przechowywać wartości liczbowe całkowite z pewnego zakresu. Jest jednak pewne ograniczenie. Nie można przecież przechowywać nieskończenie wielkich liczb w pamięci komputera, gdyż pamięć nie jest nieskończona. Z tego powodu każdy rodzaj zmiennej ma tak zwany *zakres*, czyli przedział liczbowy, jaki może przybierać. Na przykład zmienna typu `int` może zawierać wartości od -2^{31} do $2^{31}-1$. Typ określa zatem rozmiar pamięci, potrzebny do przechowania danej oraz sposób zapisu danej w pamięci komputera. W naszym przykładzie stworzona zmienna nosi nazwę `x`.

Typ danej - ***to zbiór jej możliwych wartości plus zestaw operacji, które można nad nimi wykonywać.***

Oto zestawienie typów zmiennych z zaznaczeniem ich zakresów i wielkości w bajtach:

Typ	Rozmiar	Zakres		Uwagi
	bajty	*signed*	*unsigned*	
bool	1B	przyjmuje wartość true albo false		
char	1B	-128..127 ($-2^7..2^7-1$)	0..255 ($=2^8-1$)	
wchar_t	2B	1 znak Unicode		reprezentuje znak Unicode
int	2 lub 4 B	zależnie od rozmiaru, taki jak:**short** lub **long int**	rozmiar zależy od kompilatora	
short int	2 B	-32768..32767(-2^{15} ..$2^{15}-1$)	0..65535 ($=2^{16}-1$)	
long int	4 B	-2147483648..2147483647	0..4294967295($=2^{32}-1$)	
long long int	8 B	-9223372036854775808 9223372036854775807	0.. 18446744073709551615($=2^{64}-1$)	
float	4 B	3.4e +/- 38 (7 znaków)	-	typ zmiennoprzecinkowy
double	8 B	1.7e +/- 308 (15 znaków)	-	typ zmiennoprzecinkowy
long double	8, 10 lub 12 B	1.7e +/- 308 (15 znaków)	-	typ zmiennoprzecinkowy
void			-	reprezentuje brak typu

Jeżeli użyjemy samo `long` lub `short` jest to domyślnie `long int` oraz `short int`.

Do przechowywania wartości typu prawda-fałsz w C++ wprowadzono typ `bool`, który może przybierać wartości **`true`** lub **`false`** (wszystko to są słowa kluczowe C++). Zakres zmiennych możemy dodatkowo zmodyfikować przez wprowadzenie słów kluczowych `unsigned/signed` przed typem zmiennej. Na przykład chcąc operować tylko na liczbach dodatnich dodajemy przed nazwą zmiennej słowo **`unsigned`**. Robimy to następująco:

```
unsigned int x;
```

Teraz zadeklarowana i zdefiniowana zmienna `x` ma zakres od 0 do 2^{32}.
Typ **`char`** jest, co prawda, typem całkowitym, ale jest traktowany inaczej niż typy liczbowe, gdyż w zasadzie służy do reprezentowania znaków: wartości liczbowe zmiennych tego typu odpowiadają kodom ASCII tych znaków.

Słowa kluczowe języka ***- to słowa, które mają specjalne znaczenie (np. oznaczają instrukcje sterujące lub nazwy typów prostych).***

Słowa kluczowe mogą być zarezerwowane i wtedy nie mogą być używane w innych kontekstach poza znaczeniem opisanym przez składnię języka.

```
and and_eq asm auto bitand bitor bool break case catch char class
compl const const_cast continue default delete do double
dynamic_cast else enum explicit export extern false float for friend
goto if inline int long mutable namespace new  not not_eq operator
or or_eq private  protected public register reinterpret_cast return
short signed sizeof static static_cast struct switch template this
throw true try typedef typeid typename union unsigned using virtual
void volatile wchar_t while xor xor_eq
```

Nazwy(identyfikatory) zmiennych składają się z dowolnej liczby znaków alfanumerycznych (liter i cyfr) oraz znaku podkreślenia, przy czym pierwszy znak nazwy nie może być cyfrą. Litery duże i małe są rozróżnialne; nazwa `val_X` jest inna niż np. nazwa `val_x`.

Inicjacja zmiennych.

Przy okazji deklarowania zmiennych można ustalać ich wartości, co nazywa się inicjacją.

1 sposób. Deklarację z inicjację zapisujemy w formie:

`nazwa_typu nazwa_zmiennej = wyrażenie;`

Na przyklad:

```
int a = 3;
```

co jest skróconą formą od:

```
int a;
a = 3;
```

W tym przypadku wyrażeniem inicjującym był literał. Ogólnie może być to dowolne wyrażenie. Np.

```
int a = 3;
int b = a + 1;//deklaracja zmiennej b i ustalenie wartości na 4
```

2 sposób. Drugi ze sposobów inicjacji zmiennych ma postać

`nazwa_typu nazwa_zmiennej(wartość_początkowa);`

np:

```
int x (0);
```

3 sposób. Trzeci sposób wprowadzony w standardzie C++ w 2011:

`nazwa_typu nazwa_zmiennej{wartość_początkowa};`

np.

```
int x {0};
```

```
// PRZYKŁAD: inicjalizacja zmiennych
#include <iostream>
using namespace std;

int main ()
{
  int a=5;                // inicjacja wartością : 5
  int b(3);               // inicjacja wartością : 3
  int c{2};               // inicjacja wartością : 2
  int result;             // wartość początkowa nie określona
  a = a + b;  //a=8
  result = a - c; //result=6
  cout << result;

  return 0;
}
```

Przykłady deklaracji zmiennych różnych typów:

```
#include <iostream>
using namespace std;
int main()
{
      unsigned long int ul1 = 13UL; // UL niekonieczne
      unsigned long     ul2 = 0xD;  // to samo szesnastkowo
      signed   short    ss1 = 015;  // to samo ósemkowo
      short             ss2 = 13;   // to samo
      unsigned char     aa1 = 65;
      signed   char     aa2 = 'A';
      int               aa3 = 65;
      int               aa4 = 'A';
      char              aa5 = '\101';   // 'A' ósemkowo
      char              aa6 = '\x41';   // 'A' szesnastkowo
      float k = 1.23F, m = .1F, n = 3.F;
      double x = 1.2, y = 50., z = 5e-3, v = 0.1e2;
      long double u = 1.23L, v = 30.4e-20L;
      cout << aa1 << " " << aa2 << endl
           << aa3 << " " << aa4 << endl
           << aa5 << " " << aa6 << endl;
}
```

Program ten drukuje

```
A A
65 65
A A
```

- Czasem zachodzi potrzeba wyspecyfikowania precyzyjnie typu literału liczbowego. Literał całkowity (np. `12345`) będzie zinterpretowany jako literał typu `int` (czyli `signed int`). Jeśli chcemy wymusić zinterpretowanie tego napisu jako literału typu `unsigned`, dodajemy na końcu literę '`U`' (dużą lub małą), a jeśli ma to być literał typu long, dodajemy literę '`L`'. Modyfikatory te można łączyć w dowolnej kolejności. Tak więc np. '`13UL`' jest literałem liczby 13 typu `unsigned long`.
- Literały liczbowe (całkowite) można też zapisywać w systemie ósemkowym i szesnastkowym. Aby literał był potraktowany jako zapis liczby w systemie ósemkowym, poprzedzamy go cyfrą `0` (zero). Tak np. `037` zostanie zinterpretowane jako literał typu `int` liczby o wartości dziesiętnej `38 + 7 = 31` , a `015` jako `8 + 5 = 13`. Oczywiście, w literałach ósemkowych nie mogą występować cyfry większe od siódemki. W szczególności, symbol '`\0`' oznacza znak o kodzie ASCII zero (który nazywany jest znakiem `NUL`).
- Literały w układzie szesnastkowym poprzedzamy zerem i literą '`X`' (dużą lub małą). A zatem `0x2D` to dziesiętnie `216 + 13 = 45` , a `0xD` to dziesiętnie `13`. W literałach szesnastkowych jako cyfr można użyć cyfr od `0` do `9` oraz liter od `A` do `F` (dużych lub małych).
- Literały wartości typów zmiennopozycyjnych mają standardową, wspólną dla większości języków programowania postać: można używać zapisu z kropką dziesiętną lub notacji naukowej z literą '`e`' lub, równoważnie, '`E`', pomiędzy tzw. mantysą, a wykładnikiem potęgi dziesięciu, przez którą mantysa ma być pomnożona (kropka w mantysie nie jest konieczna). Wykładnik musi być całkowity; może być poprzedzony znakiem plus lub minus. Domyślnie literał liczbowy zmiennopozycyjny jest traktowany jako literał wartości typu `double`. Jeśli, co zdarza się rzadko, chcemy wymusić traktowanie literału jako literału wartości typu `float` lub `long double`, opatrujemy go odpowiednio literą '`F`' lub '`L`'.

Wczytywanie danych z klawiatury

Wiemy już jak zadeklarować zmienną. Aby teraz z klawiatury wczytać do niej wartość będziemy korzystać z obiektu **cin** z klasy `istream`, dostępnej dzięki dołączeniu pliku `iostream`.. Jest on obiektem reprezentującym strumień informacji wchodzących do programu ze świata zewnętrznego. Źródłem tej informacji jest domyślnie klawiatura komputera. Nazwa `cin` pochodzi od *console input*. Jak `cout`, również `cin` jest obiektem, a zatem można na jego rzecz wywoływać metody zdefiniowane w jego klasie. Na razie jednak będziemy korzystać głównie z mechanizmu wyjmowania ze strumienia za pomocą operatora '»' (dwa znaki większości, bez żadnego odstępu pomiędzy nimi). Kierunek „strzałek" wskazuje na kierunek przepływu informacji, tak więc dla `cout` informacja płynie w lewo, a więc do `cout`, czyli w kierunku ekranu; dla `cin` płynie natomiast od `cin`, czyli z klawiatury.

Przyjrzyjmy się następującemu programowi:

```
//PRZYKŁAD 1
#include <iostream>
using namespace std;

int main()
{
  int x;
  cout<<"podaj wartość zmiennej x"<<endl;
  cin>>x;
  cout<<"podałeś wartość:     "<<x;

  system ("pause");
  return 0;
}
```

na początku mamy deklarację zmiennej `x` typu całkowitego `int`. Następnie po uruchomieniu programu pojawi się na ekranie monitora napis *`podaj wartość zmiennej x`* i program będzie czekał, aż wpiszemy z klawiatury liczbę całkowitą (wynik działania instrukcji `cin>>x;`). Na koniec jeszcze zostanie wyświetlone podsumowanie w postaci napisu `podałeś wartość:` i wartość zmiennej `x`, czyli wartość liczbowa, którą przed chwilą podaliśmy.

```
//PRZYKŁAD 2
#include <iostream>
#include <string>
using namespace std;

  int main()
  {
        string imie;
        int wzrost;
        double waga;
        cout << "Podaj imie, wzrost i wage: ";
        cin  >> imie >> wzrost >> waga;
        cout << imie << ", masz " << wzrost  << " cm wzrostu "
               << "i wazysz " << waga << " kg" << endl;
  }
```

Zauważmy, że:

- Każdy obiekt wyjmowany jest ze strumienia osobno (`cin>>zm1>>zm2;`)
- Wartości danych wpisywane z klawiatury muszą zgadzać się co do typu z tym, czego oczekuje program: w powyższym programie oczekiwane są dane typu `string` (napis), `int` (liczba całkowita) i `double` (liczba rzeczywista, być może z nieznikającą częścią ułamkową) i w takiej kolejności należy je wpisać.
- Poszczególne elementy danych wejściowych wpisujemy oddzielając je dowolnie długą niepustą sekwencją białych znaków (znaki odstępu, tabulacji, nowej linii, powrotu karetki - CR). Białe znaki przed pierwszą daną są ignorowane, każda następna niepusta sekwencja białych znaków jest traktowana jako separator oddzielający dane. W szczególności wynika stąd, że tą metodą nie da się wczytać napisów, które zawierają białe znaki, np. odstęp.

Wyprowadzanie danych c.d.

Zwróć uwagę, że jeśli chcemy wyświetlić jakiś komunikat, to używamy cudzysłowów, natomiast jeśli wartość danej zmiennej to już tych cudzysłowów nie stawiamy. Np. dla zmiennych `a=5` i `b=7` w wyniku działania następującego fragmentu programu:

```
cout<<"a="<<a<<endl;
cout<<"b="<<b<<endl;
cout<<"a+b="<<a+b<<endl;
```

na ekranie monitora powinniśmy otrzymać:

```
a=5
b=7
a+b=12
```

Ze zmiennych, stałych, literałów oraz wywołań funkcji, posługując się operatorami języka i nawiasami możemy konstruować **wyrażenia,** np. `a+b`.

Operacje arytmetyczne, zapisywane za pomocą operatorów, to:

- dodawanie (operator +),
- mnożenie (operator *),
- odejmowanie (operator −),
- dzielenie (/),
- reszta z dzielenia (operator %)-tylko dla liczb całkowitych
- zwiększanie o 1 (operator ++)
- oraz zmniejszanie o 1 (−−).

W przypadku liczb całkowitych operacja dzielenie jest dzieleniem całkowitoliczbowym (a więc zwraca wynik dzielenia po odrzuceniu części ułamkowej np. 1/3 „tak naprawdę" równe jest jedna trzecia, ale wynikiem tego wyrażenia będzie 0, gdyż biorą w nim udział dwie liczby całkowite i mamy tu dzielenie całkowitoliczbowe).

Operacje relacyjne i porównania. Na wartościach typów numerycznych możemy również wykonywać operacje relacyjne (porównania: <, >, <=, >=, == (czy równe?), != (czy nie równe?) i bitowe.

Operacje logiczne. Koniunkcja &&, alternatywa ||. Uwaga: Koniunkcja i alternatywa są skrótowe. Oznacza to, że prawy argument nie jest w ogóle obliczany, jeśli po obliczeniu lewego wynik jest już przesądzony.

Zadania

1. Napisz program **`Roznice.`** Program powinien pobierać od użytkownika dwie liczby całkowite `a,b` a następnie wyświetlać ich różnice, tzn `a-b` i `b-a`. Np. dla podanych wartości `a=5` i `b=7` program powinien wyświetlić:

   ```
   a-b=-2
   b-a=2
   ```

2. Napisz program **`Iloczyn`** pobierający od użytkownika dwie liczby rzeczywiste i wyświetlający ich iloczyn.

3. Napisz program **`Ilorazy`** pobierający od użytkownika dwie liczby rzeczywiste i wyświetlający ich ilorazy (a/b i b/a).

4. Napisz program **`Ilorazy całkowite.`** Pamiętaj, że iloraz całkowity możemy policzyć dla liczb całkowitych, a nie rzeczywistych.

5. Napisz program **`Reszta z dzielenia.`** Program ma wczytać z klawiatury dwie liczby a następnie policzyć resztę z dzielenia pierwszej liczby przez drugą.

6. Napisz program **`Zamiana cali na centymetry`**. Program powinien pobrać od użytkownika jedną liczbę rzeczywistą oznaczającą długość wyrażoną w calach (np. 1) po czym wydrukować na ekranie podaną długość po zmianie jednostki długości na centymetry (np. 2,54). Program powinien się ładnie przedstawić tzn. wydrukować na ekranie czytelne informacje kto i kiedy program napisał. Każda drukowana na ekranie oraz wczytywana z klawiatury liczba powinna być poprzedzona odpowiednim opisem (np. PODAJ DŁUGOŚĆ WYRAŻONĄ W CENTYMETRACH DL=).

7. Napisz program **`Zamiana złotówek na dolary`**. Program ma służyć do przeliczenia kwoty pieniędzy wyrażonej w złotówkach na kwotę dolarów. Bieżący kurs dolara znajdziesz w Internecie.

8. Napisz program przeliczający temperaturę wyrażoną w stopniach Celsjusza na temperaturę wyrażoną w skali Fahrenheita.

9. Napisz program przeliczający objętość wyrażoną w litrach na galony.

10. Napisz program przeliczający masę wyrażona w kilogramach na funty.

11. Napisz program przeliczający wielkość kąta wyrażoną w stopniach na radiany.