6. Typy danych w języku C++

W języku C++ każda zmienna, w zależności od tego, czy jest liczbą całkowitą, liczbą z częścią ułamkową, znakiem, napisem czy wartością logiczną, musi mieć jawnie przypisany odpowiedni typ danych. Określa on sposób zapisu zmiennej w pamięci komputera oraz zbiór operacji, które można wykonać na tej zmiennej.

Typy liczbowe

Wśród typów liczbowych wyróżnia się typy całkowite oraz zmiennoprzecinkowe (zmiennopozycyjne), czyli takie, które pozwalają zapisać liczby rzeczywiste. W programowaniu wartości zapisane jako 2 i 2.0 to liczby różnych typów. Pierwsza z nich jest typu całkowitego, a druga jest typu zmiennoprzecinkowego, ponieważ zawiera zapis części ułamkowej.

Nazwa typu	Przeznaczenie zmiennej	Deklaracja zmiennej w języku C++
`int`	Liczby całkowite z zakresu od –2 147 483 648 do 2 147 483 647	`int a = 4;`
`long long`	Duże liczby całkowite nie większe niż $9{,}22 \cdot 10^{18}$	`long long a = 300000000000;`
`float`	Ułamki dziesiętne i ich przybliżenia	`float a = -0.5;`
`float`	Liczby z częścią ułamkową, w których liczba cyfr znaczących (pewnych) jest nie większa niż 6	`float a = 2.45123;`
`double`	Bardzo małe liczby, które trzeba zapamiętać z większą dokładnością (nie więcej niż 15 cyfr znaczących)	`double a = 4.00000023e-12;`
`double`	Bardzo duże liczby (z częścią ułamkową) lub liczby, które trzeba zapamiętać z większą dokładnością (nie więcej niż 15 cyfr znaczących)	`double a = 4.00000023e12;`

Uwaga: Zapisy typu `e12` oraz `e-12` oznaczają notację wykładniczą (tj. odpowiednio czynniki 10^{12} oraz 10^{-12}).

Typ znakowy i typ napisowy

Pojedyncze znaki można zapisywać w zmiennych typu **`char`**. Na przykład deklaracja **`char`** `litera='a';` to deklaracja zmiennej, w której następnie zapisano literę a. Napisy, czyli tzw. łańcuchy znaków, zapisuje się z użyciem zmiennych typu **`string`**. Typ ten w sensie ścisłym nie jest typem prostym C++, ale ma charakter obiektowy, tzn. po deklaracji zmiennej uzyskujemy dostęp do wbudowanych funkcji (tzw. metod), które pozwalają łatwo wykonać operacje na napisach. Dla zmiennej zadeklarowanej jako **`string`** `napis="ABC";` liczbę znaków napisu wypiszemy na ekranie, jeśli użyjemy w kodzie zapisu **`cout`**`<<napis.size();`.

7. Podstawowe typy instrukcji w języku C++

Instrukcja warunkowa

Składnia prostej instrukcji warunkowej:

Pojedyncza instrukcja	Wiele instrukcji

`if (warunek) instrukcja;`	`if (warunek)`
	`{`
	`  instrukcja1;`
	`  instrukcja2;`
	`  ...`
	`}`

Warunek podawany po słowie **if** powinien być wyrażeniem typu logicznego (patrz dodatek 5).

Składnia instrukcji warunkowej w postaci alternatywy:

Pojedyncza instrukcja	Wiele instrukcji

`if (warunek)`
`  instrukcja;`
`else`
`  instrukcja;`
	`if (warunek)`
	`{`
	`  instrukcja1;`
	`  instrukcja2;`
	`  ...`
	`}`
	`else`
	`{`
	`  instrukcja1;`
	`  instrukcja2;`
	`  ...`
	`}`

W przypadku zagnieżdżania instrukcji warunkowych warto zadbać o odpowiednie wcięcia.

Instrukcje powtarzania. Pętla for

Składnia pętli **for** powtarzającej pojedynczą instrukcję:

```
for (licznik; warunek powtarzania; krok) instrukcja;
```

Składnia pętli **for** powtarzającej wiele instrukcji:

```
for (licznik; warunek powtarzania; krok)
{
    instrukcja1;
    instrukcja2;
    ...
}
```

Instrukcje powtarzania. Pętla `while`

Składnia pętli **`while`** powtarzającej pojedynczą instrukcję:

```
while (warunek logiczny) instrukcja;
```

Składnia pętli **`while`** powtarzającej wiele instrukcji:

```
while (warunek logiczny)
{
    instrukcja1;
    instrukcja2;
    ...
}
```

Instrukcja iteracyjna. Pętla `do while`

Składnia pętli **`do while`** powtarzającej pojedynczą instrukcję:

```
do instrukcja while (warunek logiczny);
```

Składnia pętli **`do while`** powtarzającej wiele instrukcji:

```
do
{
    instrukcja1;
    instrukcja2;
...
} while (warunek logiczny);
```

Instrukcje sterujące. Funkcje

Funkcje to podprogramy zawarte w programie głównym, które odpowiadają za realizację określonej części całego programu. Funkcje definiuje się przed definicją funkcji `main`.

Definicja funkcji składa się z nagłówka funkcji i bloku instrukcji. Nagłówek zawiera typ wartości zwracanej przez funkcję, nazwę funkcji oraz listę argumentów wraz z typami i nazwami argumentów. Budowa funkcji:

```
typ nazwa_funkcji(typ argument1, typ argument2, ...)
{
    instrukcja1;
    instrukcja2;
    ...
    return wynik;
}
```

Wartość zwracana przez funkcję jest wskazana po słowie **`return`**.
Na przykład instrukcja **`return`** `a;` oznacza, że funkcja zwróci wartość zmiennej `a`.

Gdy typem funkcji w nagłówku jest **`void`**, funkcja wykonuje zadania, które nie wymagają zwrócenia wartości wynikowej.

8. Operatory i funkcje matematyczne w języku C++

Operatory arytmetyczne

Operator	Nazwa operatora	Wyrażenie	Wynik
+	Dodawanie	`2 + 3` `2 + 3.5`	5 5.5
–	Odejmowanie	`3 - 2` `3.5 - 2`	1 1.5
*	Mnożenie	`2 * 3` `2.5 * 3.5`	6 8.75

/	Dzielenie	`3 / 2`
		`-3 / 2`
		`3.0 / 2`
			1
			–1
			1.5

%	Reszta z dzielenia (modulo)	`3 % 2`	1

W języku C++ nie ma operatora potęgowania. Kwadrat lub sześcian liczby najłatwiej zapisać jako iloczyn, odpowiednio: `2 * 2` lub `2 * 2 * 2`.
Do zapisywania potęg można też użyć funkcji pow z biblioteki `cmath`.

Wybrane funkcje matematyczne z biblioteki `cmath`

Działanie	Wyrażenie	Wynik	Uwagi
Potęgowanie	`pow(2.0; -2)`	0.25	Drugi argument funkcji to wykładnik potęgi. Może być liczbą niecałkowitą.
Pierwiastek kwadratowy	`sqrt(2.0)`	1.41421	Zwracana jest wartość przybliżenia.
Pierwiastek sześcienny	`cbrt(2.0)`	1.25992	Zwracana jest wartość przybliżenia.
Zaokrąglenie w górę (sufit)	`ceil(2.5)` `ceil(-2.5)`	3 –2	
Zaokrąglenie w dół (podłoga)	`floor(2.5)` `floor(-2.5)`	2 –3	
Zaokrąglenie (tradycyjne)	`round(2.5)` `round(-2.5)`	3 –3	
Obcięcie części ułamkowej	`trunc(2.5)` `trunc(-2.5)`	2 –2	
Wartość bezwzględna	`abs(2)` `abs(-2)`	2 2	Można też stosować funkcję `fabs` z biblioteki `math.h`
Logarytm dziesiętny	`log10(2.0)`	0.30103	Zwracana jest wartość przybliżenia.

9. Słowa kluczowe w języku C++

Słowa kluczowe nie mogą być nazwami zmiennych, funkcji ani innych obiektów w kodzie źródłowym.

Słowo kluczowe	Opis
`bool`	Nazwa typu logicznego
`const`	Używane przy definiowaniu stałej
`double`	Nazwa typu dla liczb z częścią ułamkową, o tzw. podwójnej precyzji
`else`	Używane w instrukcji warunkowej do wyodrębnienia kodu wykonywanego, gdy warunek po `if` nie jest spełniony
`false`	Stała oznaczająca wartość logiczną fałsz
`float`	Nazwa typu dla liczb z częścią ułamkową
`for`	Używane w instrukcji iteracyjnej
`if`	Używane w instrukcji warunkowej
`int`	Nazwa typu dla liczb całkowitych
`long`	Nazwa typu dla liczb całkowitych, definicja z `long long int` pozwala zapisywać w pamięci komputera liczby z większego zakresu liczb
`namespace`	Używane przy określaniu tzw. przestrzeni nazw
`return`	Używane w instrukcji, która pozwala funkcji zwrócić wartość do miejsca jej wywołania
`true`	Stała oznaczająca wartość logiczną prawda
`using`	Używane przy określaniu tzw. przestrzeni nazw
`void`	Używane w nagłówku funkcji, która nie zwraca wartości
`while`	Używane w instrukcji iteracyjnej

Inne słowa kluczowe języka C++

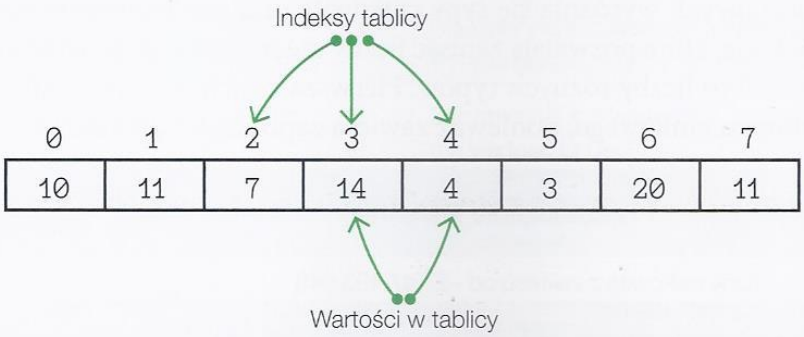
`asm`	`auto`	`break`	`case`
`catch`	`char`	`class`	`const_cast`
`continue`	`default`	`delete`	`do`
`dynamic_cast`	`enum`	`explicit`	`export`
`extern`	`friend`	`goto`	`inline`
`mutable`	`new`	`operator`	`private`
`protected`	`public`	`register`	`reinterpret_cast`
`short`	`signed`	`sizeof`	`static`
`static_cast`	`struct`	`switch`	`template`
`this`	`throw`	`try`	`typedef`
`typeid`	`type_info`	`typename`	`union`
`unsigned`	`virtual`	`volatile`	`wchar_t`

Typ logiczny

Do deklarowania zmiennych logicznych używa się typu **bool**, który przyjmuje wartości **true** i **false**. Są to słowa kluczowe języka C++. Wartość **false** reprezentowana jest w pamięci komputera przez liczbę 0, a wartość **true** – przez liczbę różną od 0.

Tablice

Tablica to struktura danych grupująca elementy określonego typu w sposób uporządkowany. Na przykład deklaracja **int** `tablica[4];` zastępuje deklarację czterech zmiennych typu **int**.



Tablice upraszczają proces zapisywania algorytmów w języku programowania, gdyż można odwołać się do wielu wartości za pomocą jednej nazwy z liczbowymi indeksami. Dostęp do poszczególnych elementów zapisanych w tablicy uzyskuje się za pomocą mechanizmu indeksowania, tj. podania pozycji elementu w tablicy, licząc od 0, np. `tablica[0]`, `tablica[1]`, `tablica[2]` i `tablica[3]`.

Zapis	Wyjaśnienie
`int tablica[4];`	Deklaracja tablicy czterech liczb typu **int**
`const int n = 4;` `int tablica[n];`	Deklaracja tablicy, której liczbę elementów określa stała n
`int tablica[3] = {1,2,3};`	Deklaracja tablicy trzech liczb typu **int** połączona z przypisaniem im wartości
`int tablica[] = {1,2,3};`	Deklaracja tablicy liczb typu **int** połączona z przypisaniem im wartości – rozmiar tablicy wyznacza się na podstawie liczby wartości
`int tablica[4] = {0};`	Deklaracja tablicy czterech liczb typu **int** połączona z przypisaniem im wszystkim wartości 0
`int tablica[4] = {1, 2};`	Deklaracja tablicy czterech liczb typu **int** połączona z przypisaniem im wartości: 1, 2, 0 i 0