Podstawy programowania w C++

Liczby – w jaki sposób komputery je widzą?

Opracował: Andrzej Nowak

Bibliografia:

CPA: PROGRAMMING ESSENTIALS IN C++ https://www.netacad.com

Czy wiesz, jak komputery wykonują obliczenia na liczbach? Być może słyszałeś o systemie binarnym i wiesz, że jest to system, którego komputery używają do przechowywania liczb i że mogą wykonywać na nich dowolną operację. Nie będziemy tutaj omawiać zawiłości systemów liczb pozycyjnych, ale powiemy, że liczby obsługiwane przez współczesne komputery są dwojakiego rodzaju:

- **liczby całkowite**, czyli liczby całkowite lub pozbawione części ułamkowej,
- **liczby zmiennoprzecinkowe** (lub po prostu zmienne), które zawierają (lub są w stanie pomieścić) część ułamkową.

Ta definicja jest dość uproszczona, ale wystarczająco dobra dla naszych celów. To rozróżnienie jest bardzo ważne, a granica między tymi dwoma typami liczb jest bardzo ścisła. Oba rodzaje liczb znacznie różnią się sposobem ich przechowywania w pamięci komputera oraz w zakresie dopuszczalnych wartości.

Typ liczby - charakterystyka liczby określającej jej:

- rodzaj,
- zasięg,
- zastosowanie.

Podstawowe dwa typy języka C ++ to:

- typ liczb całkowitych - typ integer (znany jako **`int`**) i
- typ zmiennoprzecinkowy (znany jako **`float`**).

Na razie zostawmy liczby zmiennoprzecinkowe (to prawda: więcej o nich później) i rozważmy pytanie, które na pierwszy rzut oka jest może trochę banalne:

W jaki sposób język C ++ rozpoznaje liczby całkowite?

Cóż, to prawie tak samo, jak kiedy piszesz je na kartce papieru - są po prostu ciągiem cyfr, które składają się na liczbę.

Ale jest pewien haczyk - nie możesz uwzględnić w numerze żadnych znaków, które nie są cyframi.

Przykład:

Weźmy na przykład liczbę jedenastu milionów sto jedenaście tysięcy sto jedenaście.

Gdybyś teraz wziął ołówek w ręce, prawdopodobnie napisałbyś ten numer:

```
11 111111
```

lub w celu łatwiejszego odczytu wartości – w ten sposób:

```
11 111 111
```

Na pewno ułatwia to odczytanie, jeśli liczba składa się z wielu cyfr.

Jednak C ++ widzi tę liczbę jako ciąg znaków. Musisz napisać ten numer w następujący sposób:

```
11111111
```

Jeśli tego nie zrobisz, kompilator poinformuje cię o wystąpieniu błędu.

A jak kodować liczby ujemne w C ++? Jak zwykle, dodając minus.

```
-11111111
```

Liczby dodatnie nie muszą być poprzedzone znakiem plus, ale możesz to zrobić, jeśli chcesz. Poniższe wiersze opisują tę samą liczbę:

```
+123
```

```
123
```

Zapis liczb w postaci ósemkowej i szesnastkowej

1. Zapis **liczb w postaci ósemkowej**.

Jeśli liczba całkowita jest poprzedzona cyfrą 0, będzie traktowana jako wartość ósemkowa. Oznacza to, że liczba musi zawierać cyfry z zakresu od 0 do 7.

```
0123
```

jest liczbą ósemkową z wartością dziesiętną równą 83.

2. Zapis **liczb w postaci szesnastkowej**

Taki numer powinien być poprzedzony przedrostkiem zapisanym jako 0x lub 0X.

```
0x123
```

jest liczbą szesnastkową z wartością (dziesiętną) równą 291.

Co to jest zmienna?

Zmienna – rodzaj “pojemnika” służącego do przechowywania wyników operacji na liczbach, takich jak np.:

- dodawanie,
- odejmowanie,
- mnożenie,
- dzielenie

Cechy zmiennej:

1. nazwa,

Zacznijmy od zagadnień związanych z nazwą zmiennej. Zmienne nie pojawiają się magicznie w naszym programie. My (jako programiści) decydujemy, ile i jakie zmienne chcemy mieć w naszym programie. Nadajemy im także ich nazwy.

Jeśli chcesz nadać nazwę zmiennej, musisz przestrzegać kilku ścisłych zasad:

- nazwa zmiennej może składać się z wielkich i małych liter alfabetu łacińskiego, cyfr i znaku _ (podkreślenie),
- nazwa zmiennej musi zaczynać się od litery,
- znak podkreślenia to litera (dziwna, ale prawdziwa),
- wielkie i małe litery traktowane są jako różne (trochę inaczej niż w świecie rzeczywistym - Alice i ALICE to te same imiona, ale są to dwie różne nazwy zmiennych, a co za tym idzie dwie różne zmienne).

Standard języka C ++ nie narzuca ograniczeń długości nazw zmiennych, ale określony kompilator może mieć odmienne zdanie na ten temat. Nie martw się; Zwykle ograniczenie jest tak wysokie, że jest mało prawdopodobne, że chciałbyś używać tak długich nazw zmiennych (lub funkcji).

Oto kilka poprawnych, ale nie zawsze wygodnych nazw zmiennych:

- `ja`
- `t10`
- `Kurs_wymiany`
- `licznik`
- `DaysToTheEndOfTheWorld`
- `_`

Przykłady niepoprawnych nazw zmiennych:

- ~~10 t~~ (nie zaczyna się od litery)
- ~~Adiós_Señora~~ (zawiera znaki nierozpoznawane przez kompilator)
- ~~Kurs wymiany~~ (zawiera spację)

2. typ

Typ jest atrybutem, który jednoznacznie określa, które wartości mogą być przechowywane wewnątrz zmiennej.

Poznaliśmy już typy liczb całkowitych - integer (**`int`**) i zmiennoprzecinkowych (**`float`**). Wartość zmiennej jest tym, co do niej włożyliśmy. Oczywiście można wprowadzić tylko wartość zgodną z typem zmiennej.

Tylko wartość całkowita może być przypisana do zmiennej całkowitej (lub innymi słowy, do zmiennej typu int). Kompilator nie pozwoli nam tutaj wprowadzić liczby zmiennoprzecinkowej.

3. Wartość

Pomówmy teraz o dwóch ważnych rzeczach - jak zmienne są tworzone i jak wprowadzić wartość wewnątrz nich (czy raczej - jak nadać im wartość).

Zmienna istnieje w wyniku deklaracji. Deklaracja jest strukturą składniową, która wiąże nazwę podaną przez programistę ze specyficznym typem oferowanym przez język C ++. Konstrukcja tego zgłoszenia (lub składni zgłoszenia) jest prosta – polega na użyciu nazwy pożądanego typu, a następnie podaniu nazwy zmiennej (lub zmiennych nazwy oddzielone przecinkami jeśli jest więcej niż jeden). Cała instrukcja kończy się średnikiem.

Przykład deklaracji zmiennych całkowitych:

Spróbujmy zadeklarować zmienną typu `int` o nazwie `Counter`. Odpowiednia część programu będzie wyglądać tak:

```
int Counter;
```

deklaracja kilku zmiennych tego samego typu:

```
int zmienna1, zmiany_na_koncie, faktury;
```

Nadawanie wartości zmiennym

A w jaki sposób podajemy wartość nowo zadeklarowanej zmiennej?

Musisz użyć tak zwanego operatora przypisania. Chociaż brzmi to dość tajemniczo, operator ma prostą składnię i jednoznaczną interpretację.

Operator przypisania wygląda bardzo znajomo - oto on:

```
=
```

Przykład użycia:

```
Counter = 1;
```

Powyższe stwierdzenie mówi:

przypisz wartość 1 do zmiennej o nazwie `Counter` lub, `Counter` staje się `1`

Inny przykład :

```
Wynik = 100 + 200;
```

W tym przypadku nowa wartość zmiennej `Wynik` będzie wynikiem dodania liczb 100 i 200.

Słowa kluczowe

Spójrz na obrazek poniżej - zobaczysz listę słów, które odgrywają szczególną rolę w każdym programie w języku C ++.

Są one nazywane słowami kluczowymi lub (ściślej) zastrzeżonymi słowami kluczowymi.

Są one zarezerwowane, ponieważ nie można ich używać jako nazw: ani dla zmiennych, ani dla funkcji ani żadnych innych nazwanych encji, które chcesz utworzyć.

Znaczenie słowa zarezerwowanego jest zdefiniowane i nie można go w żaden sposób zmienić.

Uwaga:

> W kompilatorze C ++ rozróżniana jest wielkość liter, można zmodyfikować dowolne z tych słów, zmieniając wielkość dowolnej litery, tworząc nowe słowo, które nie jest już zarezerwowane.

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

Komentarze

Przy tworzeniu programu dobrym zwyczajem jest dodawanie komentarza do danego fragmentu programu w celu wyjaśnienia roli parametrów i wartości zmodyfikowanych lub zwróconych jako wyniki, a także w celu wyjaśnienia, co właściwie robi dany kod.

Jak możemy zostawić coś takiego w kodzie źródłowym? Musimy to zrobić w sposób, który nie zmusi kompilatora do zinterpretowania go jako części kodu. Uwaga wstawiona do programu, która jest pominięta w momencie kompilacji, jest nazywana komentarzem.

Jeśli chcemy być precyzyjni, powinniśmy powiedzieć, że każdy komentarz jest leksykalnie równoważny jednej przestrzeni. Za każdym razem, gdy kompilator napotka komentarz w twoim programie, komentarz jest do niego całkowicie niewidoczny - z jego punktu widzenia jest to tylko jedna spacja (bez względu na to, jak długi jest prawdziwy komentarz).

Język C ++ obsługuje dwa sposoby wstawiania komentarzy:

```
// komentarze linii

/ * komentarze blokowe * /
```

Komentarz liniowy odrzuca wszystko, od miejsca, w którym znaleziono parę znaków slash (/ /) do końca tej samej linii.