Proste Tablice C++

Marek Rusinowski

25 wrzesień 2012

Spis treści

Podstawowe typy danych

Typy całkowitoliczbowe

Zmienne które mają typ całkowitoliczbowy mogą przyjmować tylko liczby całkowite. Podane ograniczenia są takie na procesorach jakie są w komputerach domowych. Pod procesorami na przykład w komórkach ograniczenia te mogą się w niektórych przypadkach nieznacznie różnić.

Typ	Ilość bajtów w pamięci	Wartość minimalna	Wartość maksymalna
char	1	-128	127
unsigned char	1	0	255
short int	2	-32768	32767
unsigned short int	2	0	65536
int	4	-2147483648	2147483647
unsigned int	4	0	4294967296
long long int	8	-9223372036854775808	9223372036854775807
unsigned long long int	8	0	18446744073709551616

Jest jeszcze typ long int aczkolwiek jest to to samo co int. Jak widać przedrostek unsigned przed nazwą typu oznacza, że zmienna o tym typie może przyjmować tylko wartości dodatnie. Jest jeszcze przedrostek signed, ale nie używa się go ponieważ każdy typ domyślnie jest signed i nie trzeba tego pisać.

Typy zmiennoprzecinkowe

Nie wnikając zbytnio w szczegóły, można powiedzieć, że typy zmiennoprzecinkowe są to typy których zmienne mogą przechowywać liczby rzeczywiste np. 1.434 albo 1.334534, ale z dokładnością tylko do określonej liczby cyfr znaczących. Na przykład jeśli jakaś liczba to 153.4673466, a typ zmiennoprzecinkowy ma dokładność 5 cyfr znaczących to liczba zapisana na komputerze to będzie 153.46.

Typ	Liczba bajtów w pamięci	Dokładność
float	4	6 cyfr
double	8	14 cyfr
long double	10	18 cyfr

Inne

Jest jeszcze dwa inne typy danych.

bool	Typ mogący w sobie przechowywać jakąś wartość logiczną, 1 albo 0, true albo false.
void	Typ oznaczający nic. Nie posiadający żadnej wartości. Nie można zrobić zmiennej tego typu, ale na przykład funkcja nie zwracająca żadnej wartości jest właśnie typu void

Zmienne

Zmienna jest to miejsce w pamięci komputera do którego przypisujemy typ i nazwę.

Definiowanie zmiennych

Zmienne definiujemy w następujący sposób.

```
typ nazwa_zmiennej;
```

na przykład

```
int x;
```

Zasięg zmiennych

Istnieje pojęcie zmiennych lokalnych i zmiennych globalnych. Zmienne możemy definiować w każdym miejscu programu czyli na przykład zaraz po załączaniu bibliotek, albo w funkcji main, lub jakiejkolwiek innej funkcji. Zmienne które zdefiniowaliśmy globalnie czyli właśnie tak luźno na przykład właśnie po załączaniu bibliotek nazywamy zmiennymi globalnymi i są one dostępne wszędzie czyli można ich używać w każdym miejscu programu. Zmienne lokalne są to zmienne definiowane w jakiejś funkcji i są one dostępne tylko w niej. Nie można odwoływać się do zmiennych zdefiniowanych w main'ie z jakiejś innej funkcji.

Operator przypisania

Do zmiennej da się przypisać pewną wartość, stosując *operator przypisania* ma to formę

```
zmienna = wartość;
```

Oznacza to, że zmiennej po lewej stronie znaku '=' jest przypisana wartość po jego prawej stronie. Operator przypisania można stosować także przy definiowaniu zmiennej.

Przykłady

```
int x = 5;
```

```
x = -6;
x = 203;
char c;
c = 'a';
double f = 4.344233;
bool b = true;
```

Operatory

Operatory arytmetyczne

Są to operatory które powodują pewne działanie arytmetyczne. W C++ kolejność wykonywania działań arytmetycznych jest taka sama jak w matematyce.

Operator	Przykład	Opis
+	`int x = 3 + 3 + 7;` `int v = x + (-5);`	Zwraca sumę liczb
-	`int x = 3 + 3 - 7;` `int v = x - 5;`	Zwraca różnicę liczb
*	`int x = 2 * 4;` `int v = x * 5;`	Zwraca iloczyn liczb

/	`int x = 2 / 2;`
	`int f = (4 * x) / 5;`
	`double f = 3.0 / 4.0;`
		Zwraca iloraz dwóch liczb. Gdy liczby są całkowite zwraca wynik taki jak przy dzieleniu z resztą na przykład 5 / 2 = 2.

%	`int x = 5 % 2;`	Zwraca resztę z dzielenia liczb.

Operatory porównywania

Operatory porównywania służą do porównywania dwóch wartości. Zwracają one wartość logiczną którą można przypisać do zmiennej typu bool albo użyć w instrukcji sterującej if.

Operator	Przykład	Opis
>	`3 > 1`	Większe
<	`1 < 3`	Mniejsze
<=	`1 < 3` `1 <= 1`	Mniejsze lub równe
>=	`3 > 1` `3 >= 3`	Większe lub równe

`==`	`4 == 4`	Równe
`!=`	`5 != 4`	Nie równe

Operatory logiczne

Operatory logiczne zwracają wartość logiczną na podstawie dwóch (lub jednej) wartości logicznych.

`||`	`a || b`	Lub
		`true || true = true`
		`true || false = true`
		`false || true = true`
		`false || false = true`

`&&`	`a && b`	Oraz
		`true && true = true`
		`true && false = false`
		`false && true = false`
		`false && false = false`

`!`	`!a`	Negacja
		`!true = false`
		`!false = true`

Inne

Jest jeszcze wiele innych operatorów. Najczęściej stosowanymi innymi operatorami są operator inkrementacji oraz dekrementacji (inkrementacja zwiększanie o 1, dekrementacja zmniejszanie o 1)

```
int x = 5;
++x;
int v = x;
--x;
```

Po wykonaniu tego kawałka kogu w zmiennej v będzie wartość 6, a w zmiennej x wartość 5. ‘++’ po prostu zwiększa wartość zmiennej o 1 a ‘--’ zmniejsza wartość zmiennej o 1.

Podstawowe konstrukcje sterujące

Konstrukcje sterujące kontrolują wykonanie programu. Domyślnie kod wywołuję się od góry do dołu, dzięki konstrukcją sterującym, możemy decydować czy wykonamy jakiś fragment kodu czy nie.

Instrukcje warunkowe

Podstawową instrukcją warunkową jest instrukcja if

```
if (wyrażenie_logiczne) {
  troche kodu
}
```

Oznacza to, że jeżeli wyrażenie logiczne ma wartość true to blok kodu między nawiasami się wykona na przykład

```
int x = 4;
if (x < 10) {
  x = 20;
}
```

Po wykonaniu tego fragmentu x będzie miał wartość 20 bo wartośc wyrażenia logicznego w nawiasie jest prawdziwa. Po if można dać instrukcję else

```
if (warunek logiczny) {
  kod wykonujący się gdy warunek logiczny jest spełniony
} else {
  kod wykonujący się gdy warunek logiczny nie jest spełniony
}
```

Pętle

Są dwie pętle, które są sobie równoważne. Jedną można zapisać za pomocą drugiej. Pętla jest to taki if tylko, że kod wykonujący się w jego wnętrzu jest wykonywany w kółko dopuki warunek logiczny jest prawidziwy. Jeśli warunek logiczny jest zawsze spełniony program utyka w tej pętli i program się zawiesza.

Pętla while

```
while (warrunek logiczny) {
  kod
}
```

Pętla for

```
for (wyrażenie1; wyrażenie logiczne; wyrażenie2) {
  kod
}
```

Taką pętlę for można zapisać za pomocą pętli while, i kod będzie równoważny

```
wyrażenie1;
while (wyrażenie logiczne) {
  kod
  wyrażenie2;
}
```

Funkcje

Funkcję intuicyjnie są podprogramami które mogą zostać wywołane w każdym momencie działania programu z każdego jego miejsca, nawet z siebie samej.

Definiowanie funkcji

Funkcje definiuje się w następujący sposób

```
zwracany_typ nazwa_funckji (lista parametrów) {
  KOD FUNKCJI
}
```

Na przykład, to funkcja przyjmująca za parametry dwie liczby i zwracająca ich sumę.

```
int dodaj (int a, int b) {
  int c = a + b;
  return c;
}
```