

Definicja typu

Zacznijmy od tego - co to jest typ. Można powiedzieć, że typ odwzorowuje nam pewien zbiór wartości. Przykładowo - `int` to typ, za którego pomocą można zapisać 32-bitową liczbę całkowitą (na standardowych architekturach - przykładowo, w 16-bitowym Windowsie `int` miał wielkość 16 bitów). Z większością typów w języku C++ możemy używać *modyfikatorów*, które w jakiś sposób nam zmieniają ten typ. Na przykład `unsigned` powoduje że typ staje się typem *bez znaku* - czyli, przykładowo, z użyciem `unsigned int`, możemy zapisać 32-bitową liczbę całkowitą nieujemną (od 0 do $2^{32} - 1$). Poniżej opiszę listę najczęściej wykorzystywanych typów *fundamentalnych*, czyli podstawowych, oraz ich właściwości i modyfikatory.

Podstawowe typy

Podstawowe 6 typów w języku C++ to

`void`

`void` jest typem pustym. To znaczy, że jego zbiór wartości jest pusty - czyli nie możemy za jego pomocą (teoretycznie) zapisać żadnej wartości. Najczęściej wykorzystywany w definicjach funkcji, na przykład `void f();`, oznacza że funkcja `f` nie zwróci żadnej wartości.

`bool`

`bool` odwzorowuje nam wartość logiczną - nazwa wzięła się od `boolean`. Przy jej użyciu możemy zapisać dwie wartości - `true` albo `false`. Tutaj warto zaznaczyć, że wszystkie operacje porównania (z użyciem operatorów porównań, na przykład `<` lub `==`) zwracają wartość typu `bool`.

`char`

`char` to specyficzny typ danych o dość szerokim zastosowaniu. Standardowo jest to typ za pomocą którego możemy zapisać znak ASCII (oraz UTF-8, od C++ 14). Może być on jednocześnie traktowany jako liczba 8-bitowa. Tablice typu `char` są zwane *łańcuchami znaków*, na przykład `"Hello world!"`.

`int`

`int` to chyba najpowszechniej wykorzystywany typ danych. W aktualnych architekturach jest to standardowo 32-bitowy typ, dzięki któremu możemy zapisywać liczby całkowite. Jest on wystarczająco duży dla większości operacji matematycznych.

`float`

`float` to 32-bitowy typ zmiennoprzecinkowy o stosunkowo małej precyzji. Pozwala nam na zapisywanie wartości z około siedmioma cyframi (na przykład 123.4567).

`double`

`double` to 64-bitowy typ zmiennoprzecinkowy o *podwójnej precyzji* (w porównaniu do `float`'a). Pozwala na zapisywanie wartości z około 15 cyframi.

Modyfikatory typu `int`

Typ `int` jest podstawowym typem dla liczb całkowitoliczbowych i z tego względu możemy nieco go zmodyfikować. Najbardziej podstawowymi modyfikatorami są `signed` i `unsigned`. Domyślnie, każdy typ zachowuje się tak, jakby miał modyfikator `signed` - czyli jest w stanie reprezentować liczby całkowite dodatnie i ujemne. Modyfikatora `unsigned` możemy chcieć użyć w kilku sytuacjach:

- **Jesteśmy w 100% pewni że nie potrzebujemy ujemnych wartości i nie mają prawa takowe się pojawić, a przyda nam się więcej wartości dodatnich jakie możemy zapisać przy użyciu typu** - Typ liczbowy o wielkości n bitów ze znakiem pozwala nam na zapisywanie w nim wartości w zakresie około $[-(2^{n-1}), 2^{n-1}]$ (zależnie od konwencji). Dzieje się tak dlatego, że według bitowej konwencji zapisu liczb ze znakiem, ostatni (najstarszy) bit jest bitem znaku. Kiedy dajemy typowi modyfikator `unsigned`, ten bit może zostać wykorzystany do zapisu liczby, co powoduje zmianę zakresu na $[0, 2^n - 1]$ - zyskujemy 2 razy większy zakres dla dodatnich liczb, kosztem ujemnych.
- **Chcemy mieć typ który pozwala nam na bezpieczne operacje bitowe** - Typy ze znakiem, z racji że są zapisywane w innej konwencji, nie do końca nadają się do operacji bitowych. Jeśli potrzebujemy typów *strict* do operacji na bitach, powinniśmy użyć modyfikatora `unsigned`.

W każdej innej sytuacji, sugeruję nie używać modyfikatora `unsigned`, bo możemy sobie zaszkodzić, zamiast pomóc.

Notka: te modyfikatory działają również z typem `char`. `unsigned char` to odpowiednik typu `byte` znanego z innych języków, służącego do przechowywania 8-bitowych wartości bez znaku, czyli po prostu czystych danych.

Mamy też do dyspozycji modyfikatory `long` i `short`. Modyfikator `short` spowoduje że `int` zostanie zoptymalizowany pod względem pamięci, przez co stanie się typem *co najmniej* 16-bitowym. Modyfikator `long` spowoduje że `int` będzie miał wielkość *co najmniej* 32 bitów - z racji że na większości architektur teraz i tak ma taką wielkość, to nie robi to większej różnicy. Natomiast możemy jeszcze użyć modyfikatora `long long` który spowoduje że wielkość naszego `inta` będzie wynosiła *co najmniej* 64 bity.

Notka: Istnieje również typ `long double`, który jest standardowo 80-bitowym (!) typem danych, ale nie zalecam używania go bez powodu.

Jak używać tych modyfikatorów

Wystarczy dodać je w dowolnej kolejności do typu. Na dodatek, możemy pominąć `int` jeśli używamy któregoś z modyfikatorów - kompilator i tak uzna że używamy `inta`. Na przykład

```
long long int very_long_int;
unsigned unsigned_int;
short short_int;
```

Następna część - Operatory