

Ravesli [Ravesli](#)

- [Уроки по C++](#)
- [OpenGL](#)
- [SFML](#)
- [Qt5](#)
- [RegExr](#)
- [Ассемблер](#)
- [Купить .PDF](#)



Урок №30. Размер типов данных

[Юрий](#) |

- [Уроки C++](#)

|

Обновл. 2 Сен 2020 |

50275

[13](#)

Как мы уже знаем из [урока №28](#), память на современных компьютерах, как правило, организована в блоки, которые состоят из байтов, причем каждый блок имеет свой уникальный адрес. До этого момента, память можно было сравнивать с почтовыми ящиками (с теми, которые находятся в каждом подъезде), куда мы можем поместить информацию и откуда мы её можем извлечь, а имена переменных — это всего лишь номера этих почтовых ящиков.

Тем не менее, эта аналогия не совсем подходит к программированию, так как переменные могут занимать больше 1 байта памяти. Следовательно, одна переменная может использовать 2, 4 или даже 8 последовательных адресов. Объем памяти, который использует переменная, зависит от типа данных этой переменной. Так как мы, как правило, получаем доступ к памяти через имена переменных, а не через адреса памяти, то компилятор может скрывать от нас все детали работы с переменными разных размеров.

Есть несколько причин по которым полезно знать, сколько памяти занимает определенная переменная/тип данных.

Во-первых, чем больше она занимает, тем больше информации сможет хранить. Так как каждый бит содержит либо 0, либо 1, то 1 бит может иметь 2 возможных значения.

2 бита могут иметь 4 возможных значения:

бит 0 бит 1

0 0

0 1

1 0

1 1

3 бита могут иметь 8 возможных значений:

бит 0 бит 1 бит 2

0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

По сути, переменная с n-ным количеством бит может иметь 2^n возможных значений. Поскольку байт состоит из 8 бит, то он может иметь 2^8 (256) возможных значений.

Размер переменной накладывает ограничения на количество информации, которую она может хранить. Следовательно, переменные, которые используют больше байт, могут хранить более широкий диапазон значений.

Во-вторых, компьютеры имеют ограниченное количество свободной памяти. Каждый раз, когда мы объявляем переменную, небольшая часть этой свободной памяти выделяется до тех пор, пока переменная существует. Поскольку современные компьютеры имеют много памяти, то в большинстве случаев это не является проблемой, особенно когда в программе всего лишь несколько переменных. Тем не менее, для программ с большим количеством переменных (например, 100 000), разница между использованием 1-байтовых или 8-байтовых переменных может быть значительной.

Размер основных типов данных в C++

Возникает вопрос: «Сколько памяти занимают переменные разных типов данных?». Вы можете удивиться, но размер переменной с любым типом данных зависит от компилятора и/или архитектуры компьютера!

Язык C++ гарантирует только их минимальный размер:

Категория	Тип	Минимальный размер
Логический тип данных	bool	1 байт
Символьный тип данных	char	1 байт
	wchar_t	1 байт
	char16_t	2 байта
	char32_t	4 байта
Целочисленный тип данных	short	2 байта
	int	2 байта
	long	4 байта
	long long	8 байт
Тип данных с плавающей запятой float		4 байта

double 8 байт

long double 8 байт

Фактический размер переменных может отличаться на разных компьютерах, поэтому для его определения используют оператор `sizeof`.

Оператор `sizeof` — это унарный оператор, который вычисляет и возвращает размер определенной переменной или определенного типа данных в байтах. Вы можете скомпилировать и запустить следующую программу, чтобы выяснить, сколько занимают разные типы данных на вашем компьютере:

```
1  #include <iostream>
2
3  int main()
4  {
5      std::cout << "bool:\t\t" << sizeof(bool) << " bytes" << std::endl;
6      std::cout << "char:\t\t" << sizeof(char) << " bytes" << std::endl;
7      std::cout << "wchar_t:\t" << sizeof(wchar_t) << " bytes" << std::endl;
8      std::cout << "char16_t:\t" << sizeof(char16_t) << " bytes" << std::endl;
9      std::cout << "char32_t:\t" << sizeof(char32_t) << " bytes" << std::endl;
10     std::cout << "short:\t\t" << sizeof(short) << " bytes" << std::endl;
11     std::cout << "int:\t\t\t" << sizeof(int) << " bytes" << std::endl;
12     std::cout << "long:\t\t\t" << sizeof(long) << " bytes" << std::endl;
13     std::cout << "long long:\t" << sizeof(long long) << " bytes" << std::endl;
14     std::cout << "float:\t\t\t" << sizeof(float) << " bytes" << std::endl;
15     std::cout << "double:\t\t" << sizeof(double) << " bytes" << std::endl;
16     std::cout << "long double:\t" << sizeof(long double) << " bytes" << std::endl;
17     return 0;
18 }
```

Вот результат, полученный на моем компьютере:

bool:	1 bytes
char:	1 bytes
wchar_t:	2 bytes
char16_t:	2 bytes
char32_t:	4 bytes
short:	2 bytes
int:	4 bytes
long:	4 bytes
long long:	8 bytes
float:	4 bytes
double:	8 bytes
long double:	8 bytes

Ваши результаты могут отличаться, если у вас другая архитектура, или другой компилятор. Обратите внимание, оператор `sizeof` не используется с типом `void`, так как последний не имеет размера.

Если вам интересно, что значит `\t` в коде, приведенном выше, то это специальный символ, который используется вместо клавиши TAB. Мы его использовали для выравнивания столбцов. Детально об этом мы еще поговорим на соответствующих уроках.

Интересно то, что `sizeof` — это один из 3-х операторов в языке C++, который является словом, а не символом (еще есть `new` и `delete`).

Вы также можете использовать оператор `sizeof` и с переменными:

```
1 #include <iostream>
2
3 int main()
4 {
5     int x;
6     std::cout << "x is " << sizeof(x) << " bytes" << std::endl;
7 }
```

Результат выполнения программы:

x is 4 bytes

На следующих уроках мы рассмотрим каждый из фундаментальных типов данных языка C++ по отдельности.

Оценить статью:

★★★★★ (411 оценок, среднее: 4,94 из 5)

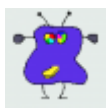


[← Урок №29. Тип данных void](#)



[Урок №31. Целочисленные типы данных: short, int и long →](#)

Комментариев: 13



1. *Sagynysh:*

[2 мая 2020 в 21:06](#)

а существует ли тип данных `long float`?

[Ответить](#)



1. *Михаил:*

[12 августа 2020 в 14:22](#)

Да. Существует. В данном случае символ `Long` означает — Длинное. В итоге получается то же самое, что и `Double`. Кстати. Изначально `Long Double` обязан был быть 10 байтным вещественным числом по формату FPU.

[Ответить](#)2. *Артеми́й:*[20 января 2020 в 13:02](#)

В настройках компилятора надо сделать поддержку c++11

[Ответить](#)3. *Сергей:*[16 ноября 2019 в 22:15](#)

Почему нельзя так?

```

1 using namespace std;
2 cout << sizeof(int) << endl;

```

[Ответить](#)1. *Sasha:*[8 января 2020 в 20:37](#)

в любой программе должна быть функция main, где она у тебя? еще библиотека iostream нужна для cout

[Ответить](#)1. *AdamBeno:*[7 февраля 2020 в 02:09](#)

Да ну нах, не может быть.

[Ответить](#)4. *Дороу:*[23 августа 2019 в 12:03](#)

clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/final)

☐ clang++-7 -pthread -o main io.cpp main.cpp

☐ ./main

bool: 1 bytes

char: 1 bytes

wchar_t: 4 bytes

char16_t: 2 bytes

char32_t: 4 bytes

short: 2 bytes

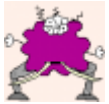
int: 4 bytes

long: 8 bytes

long long: 8 bytes

float: 4 bytes
double: 8 bytes
long double: 16 bytes

[Ответить](#)



5. *Дмитрий:*
[20 августа 2019 в 11:16](#)

ОС Линукс Роса.
Компилятор выдаёт ошибку

```
||=== Build: Debug in typesize (compiler: GNU GCC Compiler) ===|
/home/dima/projects/typesize/main.cpp|9|ошибка: нет декларации «char16_t» в этой области
видимости|
/home/dima/projects/typesize/main.cpp|10|ошибка: нет декларации «char32_t» в этой области
видимости|
||=== Build failed: 2 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ===|
```

Может, я какую библиотеку не подключил? До этого всё получалось хорошо, даже `uniform` инициализация прошла, правда с предупреждением,

[Ответить](#)



6. *natovan:*
[7 декабря 2018 в 20:00](#)

Странно, у меня `long` и `long long` имеют 8 байт. Получается, использовать `long long` нет смысла.

[Ответить](#)



1. *Бубушян:*
[22 апреля 2019 в 12:45](#)

Чем больше `long`, тем больше вес болта, смекаешь, дружище

[Ответить](#)



1. *Илья:*
[15 июня 2019 в 13:49](#)

Встретимся на экзамене, шутник

[Ответить](#)



2. *Проггер:*
[24 августа 2019 в 20:06](#)

У `long long` большой диапазон величины значения которое он может принимать (можно использовать для операций с большими числами).

Прошу поправить если не прав.

[Ответить](#)



1. *Sasha:*

[8 января 2020 в 20:40](#)

В уроке сказано, что все зависит от компилятора. В данном случае long и long long занимают 8 байт, следовательно диапазон значений тоже одинаковый. То есть нет разницы long или long long. НО это только у автора комментария, у меня например long 4, а long long 8

[Ответить](#)

Добавить комментарий

Ваш E-mail не будет опубликован. Обязательные поля помечены *

Имя *

Email *

Комментарий

☐ Сохранить моё Имя и E-mail. Видеть комментарии, отправленные на модерацию





☐ Получать уведомления о новых комментариях по электронной почте. Вы можете [подписаться](#) без комментирования.

[TELEGRAM](#)  [КАНАЛ](#)



[ПАБЛИК](#) 

ТОП СТАТЬИ

-  [Словарь программиста. Сленг, который должен знать каждый кодер](#)
-  [Урок №1. Введение в программирование](#)
-  [70+ бесплатных ресурсов для изучения программирования](#)
-  [Урок №1: Введение в создание игры «Same Game»](#)

-  [Урок №4. Установка IDE \(Интегрированной Среды Разработки\)](#)

- [Ravesli](#)
- - [О проекте](#) -
- - [Пользовательское Соглашение](#) -
- - [Все статьи](#) -
- Copyright © 2015 - 2020