

РАЗРАБОТКА НА C++

Урок 2.

Переменные и типы данных

План

- Что такое тип данных?
- Почему важно в C++ знать типы данных?
- Основные типы данных в C++
- Что такое переменные?
- Примеры объявления переменных разных типов.
- Определение переменных.
- Правила именования переменных.
- Хорошие практики по именованию переменных (например, использование осмысленных имен).
- Как лучше всего сдавать домашнее задание?
- Домашнее задание

РАЗРАБОТКА НА C++

Типы данных в C++

Единицы измерения памяти

Сейчас современные компьютеры уже не так зависят от объема памяти, но для понимания работы типов данных нам важно понимать что такое байт и бит:

бит хранит в себе либо 1 либо 0 (есть сигнал - нету сигнала)

байт хранит 8 бит (например 00110110 или 10010000) и может кодировать информацию

Название	Условное обозначение	Соотношение с другими единицами
Байт	Байт	1 Байт = 2^3 бит = 8 бит
Килобит	Кбит	1 Кбит = 2^{10} бит = 1024 бит
Килобайт	Кбайт (Кб)	1 Кб = 2^{10} Байт = 1024 Байт
Мегабайт	Мбайт (Мб)	1 Мб = 2^{10} Кб = 1024 Кб
Гигабайт	Гбайт (Гб)	1 Гб = 2^{10} Мб = 1024 Мб
Терабайт	Тбайт (Тб)	1 Тб = 2^{10} Гб = 1024 Гб

Что такое тип данных?

Это спецификация, которая определяет, какой вид данных может быть сохранен в переменной. Он определяет, как компьютер будет интерпретировать и обрабатывать данные, которые хранятся в этой переменной.

Данные всегда занимают место в памяти и чаще всего их объем исчисляется в байтах (число $7_{10} = 00000111_2 = 2^2 + 2^1 + 2^0 = 4 + 2 + 1$)

В C++ типы данных должны быть определены для всех функций и переменных до того как программа будет скомпилирована. **Язык C++ является строго типизированным** (статически типизированным)

Почему важно в C++ знать типы данных?

Корректность данных: если вы храните возраст человека, вы можете использовать целочисленный тип данных (например, `int`), потому что возраст обычно измеряется целыми числами. Если вы храните десятичные доли, вы бы использовали тип данных с плавающей запятой и так далее.

Эффективность памяти: если вам нужно хранить целые числа в диапазоне от -100 до 100, то можно использовать тип данных `char` или `short`, чтобы сэкономить память по сравнению с `int`.

Безопасность и устойчивость кода: если вы пытаетесь сохранить текстовую строку в переменной типа `int`, это приведет к ошибке, и компилятор предупредит вас о некорректном использовании.

Читаемость кода: типы данных делают ваш код более читаемым и понятным для других программистов, работающих над проектом, а также для вас самого в будущем.

Поддержка разных операций: Разные типы данных поддерживают разные операции. Например, с целочисленными типами можно выполнять математические операции, а с символьными типами - операции со строками и символами.

Основные типы данных в C++

int (целые числа): Используется для хранения целых чисел, например, -5, 0, 42. Как правило на современных архитектурах ПК занимает 4 байта

00000000 00000000 00000000 00000000 это число 0

если бы мы не учитывали отрицательные числа то:

11111111 11111111 11111111 11111111 равнялось бы числу 4 294 967 295

и такой тип данных действительно существует называется **unsigned int** (т. е. беззнаковый - не даст хранить отрицательные значения)

Основные типы данных в C++

Но на самом деле, в типе данных `int` самый первый(старший) бит - это знак минус или плюс:

Минимальное число:

10000000 00000000 00000000 00000000 это число -2 147 483 648

Максимальное число:

01111111 11111111 11111111 11111111 это число 2 147 483 647

Набор бит *только из единиц* это самое большое целое отрицательное число, т.е. число -1

Основные типы данных в C++

Разные модификаторы могут определять диапазон значений, например:

int для обычных целых чисел (*чаще всего от $-2\,147\,483\,648$ до $2\,147\,483\,647$)

short int для коротких целых чисел (от -32768 до 32767)

long int для длинных целых чисел (*от $-9\,223\,372\,036\,854\,775\,808$ до $+9\,223\,372\,036\,854\,775\,807$)

unsigned int для беззнаковых чисел (от 0 до $4\,294\,967\,295$)

есть еще варианты целых чисел: **unsigned short**, **long long**, **unsigned long long**.

Основные типы данных в C++

Если мы попытаемся к максимальному значению типа данных прибавить число, то произойдет “переполнение типа данных”, т.е. биты, которые не убралось в двоичном представлении числа, просто будут упущены и мы увидим например такой результат:

$2147483647 + 1 = -2147483648$ (для int)

$4294967295 + 1 = 0$ (для unsigned int)

если к MAX значению прибавить 1, то мы получим MIN значение и наоборот

История из видеоигр

Ганди (или Mahatma Gandhi) был одним из лидеров играбельных цивилизаций в серии Civilization. В игре у каждого лидера были определены атрибуты и характеристики, включая агрессивность, которая оценивалась от 1 до 10. Ганди изначально имел низкую агрессивность, что делало его одним из самых миролюбивых лидеров в игре.

Однако в одной из версий Civilization была ошибка в коде, которая при прогрессе игры и использовании определенных технологий приводила к тому, что уровень агрессивности Ганди становился отрицательным. **А так как тип данных не мог хранить числа ниже 0, то из-за “переполнения” агрессивность становилась максимально большой.**



Основные типы данных в C++

float: представляет вещественное число с плавающей точкой в диапазоне +/- 3.4E-38 до 3.4E+38. В памяти занимает 4 байта (32 бита)

double: представляет вещественное число двойной точности с плавающей точкой в диапазоне +/- 1.7E-308 до 1.7E+308. В памяти занимает 8 байт (64 бита)

E - это умножить на 10

E+38 - умножить на 10^{38}

E-38 - умножить на 10^{-38} или по другому умножить на $(0.1)^{38}$

Основные типы данных в C++

char(символ, закодированный числом):
представляет один символ в кодировке ASCII (. Занимает в памяти 1 байт (8 бит).
Может хранить любое значение из диапазона от -128 до 127, либо от 0 до 255

ASCII Table

Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char	Dec	Hex	Oct	Char
0				32			[space]	64			@	96			`
1				33			!	65			A	97			a
2				34			"	66			B	98			b
3				35			#	67			C	99			c
4				36			\$	68			D	100			d
5				37			%	69			E	101			e
6				38			&	70			F	102			f
7				39			'	71			G	103			g
8				40			(72			H	104			h
9				41)	73			I	105			i
10				42			*	74			J	106			j
11				43			+	75			K	107			k
12				44			,	76			L	108			l
13				45			-	77			M	109			m
14				46			.	78			N	110			n
15				47			/	79			O	111			o
16				48			0	80			P	112			p
17				49			1	81			Q	113			q
18				50			2	82			R	114			r
19				51			3	83			S	115			s
20				52			4	84			T	116			t
21				53			5	85			U	117			u
22				54			6	86			V	118			v
23				55			7	87			W	119			w
24				56			8	88			X	120			x
25				57			9	89			Y	121			y
26				58			:	90			Z	122			z
27				59			;	91			[123			{
28				60			<	92			\	124			
29				61			=	93]	125			}
30				62			>	94			^	126			~
31				63			?	95			_	127			

Основные типы данных в C++

char по своей сути хранит число, но компилятор при работе с ним использует таблицу чтобы узнать какой символ закодирован этим числом. Поэтому примеры char следующие:

'g' - это символ (указываем что это **char** одинарными кавычками)

'1' - это символ, так как находится в одинарных кавычках

49 - это код символа '1'

Основные типы данных в C++

string: тип данных, позволяющий хранить последовательность char (в будущем узнаем что такое “массив”)

Строка обозначается двойными кавычками, например:

“Hello World!” - строка может хранить любые символы в зависимости от кодировки программы.

“1” - это тоже строка, так как мы добавили двойные кавычки

“g” - это уже не символ, а строка

Основные типы данных в C++

bool: Логический тип данных, который может хранить либо правду, либо ложь (0 или 1). Чаще всего используется при проверках, о которых будем говорить на следующих вебинарах.

РАЗРАБОТКА НА C++

Переменные в C++

Переменные в C++

Это именованная область памяти, которая используется для хранения и управления данными. Переменные позволяют программистам сохранять значения, с которыми они работают, и манипулировать ими в процессе выполнения программы.

Тип_данных Имя_переменной = Значение;

```
int age = 16;
```

также можно объявить пустую переменную, например:

```
int x;
```

Важно! Тип данных для переменной нужно указывать лишь один раз

Точка с запятой и комментарии в коде

В конце почти каждой операции в C++ **необходимо ставить символ “;”** (точка с запятой), так программа понимает, что инструкция завершена.

В C++ есть возможность оставлять комментарии. Компилятор не будет учитывать текст который находится после символов “//”:

// текст; int x = 5; ничего из этого компилятор не увидит

// и это тоже

Также можно ставить комментарий сразу на много строк “/**/”

/* тут много строк кода

тут всё ещё комментарии

только здесь они заканчиваются */

Ошибки в объявлении переменных C++

Если мы не объявим тип данных, то компилятор выдаст ошибку:

`int x = 3` // ошибка, забыли точку с запятой

`y = 3;` // ошибка, если заранее не написать `int y;`

`3;` // хоть и не ошибка, но смысла в такой записи не будет :)

`3 = y;` // ошибка, не путайте местами, такая запись не сработает

`integer y = 3;` // ошибка, слово `integer` в C++ не используется

`INT y = 3;` // ошибка, регистр букв имеет значение

`int 3x = 3;` // ошибка, имя не может начинаться с цифры

Правильные варианты объявления переменных

`int x = 3;` // в переменную x записали число 3

`x = 5;` // перезаписали, теперь x равно числу 5

`x = 13-5;` // можно записывать результаты вычислений в переменные

`int y = -3;` // запись отрицательного числа

`x = y+y;` // $-3 + (-3) = -6$ записываем -6 в переменную x (y не изменится)

`int z = x;` // записываем то же значение, что хранится в переменной x

`z = y-z;` // $y = -3; z = -6 \Rightarrow -3 - (-6) = 3$, значит z станет равным 3

Названия переменных

Идентификатором называется последовательность цифр и букв, а также специальных символов, при условии, что первой стоит буква или специальный символ

Правила составления идентификаторов:

- Начинается с буквы или знака «_» (“x” и “_x” это два разных допустимых имени)
- Регистр букв имеет значение (“x” и “X” это два разных допустимых имени)
- Может иметь любое количество символов, но значимыми являются только первые 31 идентификатор не должен совпадать с ключевыми словами, с ключевыми словами и именами функций библиотеки компилятора языка программирования C++

Ключевые слова

Некоторые слова нельзя использовать в качестве названия переменных, например:

asm, break, case, catch, continue, default, do, else, for, goto, if, new, return, sizeof, switch, throw, try, typedef, typeid, while, char, double, float, int, void, long, short, signed, unsigned и т.д.

Эти слова зарезервированы языком C++ для выполнения тех или иных операций или обозначают типы данных.

Но названия по типу “int1” или “case_one” использовать можно, так как они отличаются от ключевых

Хорошие практики для именования переменных

Важно понимать как вы называете переменные, так как читаемость кода и отладка существенно упрощаются, когда имена осмыслены. Пример:

`p`, `e1`, `p_i`, `a` - угадайте что эти переменные означают в игре?

игрок, первый враг, инвентарь игрока, патроны

лучше будет написать всё-таки так:

`player`, `enemy_1`, `playerInventory`, `ammo`

Три распространенных стиля кода для переменных:

camelCase - первое слово с маленькой, последующие с большой буквы

PascalCase - все слова с большой буквы

snake_case - все слова с маленькой буквы и разделены нижним подчеркиванием

Хорошие практики для именования переменных

Для кода использующего много вычислений, например для нахождения дискриминанта, переменные x , y , z , $d1$, $a1$ и т.д. являются хорошими названиями, так как они пришли к нам из математики и человек читающий наш код сможет легко понять что мы имеем в виду.

Также часто используемые слова можно сократить до нескольких букв

hp - HitPoints

exp - Experience

dps - DamagePerSecond

Как сдавать домашние задания?

Лучшим вариантом будет создать в программе Word или в Google Documents документ (.doc или .docx) и скопировать в него весь код из файла, в котором вы писали код. Также необходимо прикрепить после кода скрины того, что ваша программа работает и выдает результаты соответствующие заданию.

Все комментарии к коду или вопросы можно написать как внутри файла, так и в сообщении на платформе Ims при сдаче домашней работы

Весь проект в архиве сдавать не нужно, пока это не будет указано в домашнем задании!