Типы данных и стандартный ввод-вывод

Программный объект

Во время разработки требуется создавать, получать и сохранять данные. Под данными подразумеваются числа, строки и другие сущности, необходимые для решения задач

Объект - место в памяти, имеющее тип, определяющий вид информации, разрешенной для хранения.

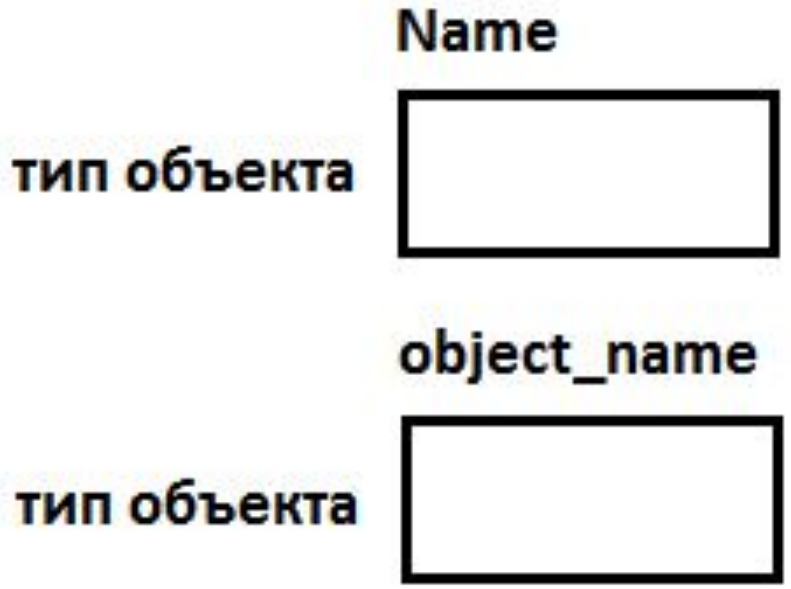
Переменная - именованный объект.

Определение переменной

- инструкция, вводящая новое имя в программе и выделяющая память для переменной, называется *определением* (definition), не путать с *объявлением* (declaration);
- в языке C++ определение переменной обычно происходить одновременно с объявлением;
- C++ поддерживает возможность определения переменных в коде программы по мере необходимости.

Синтаксис определения переменной:

тип_переменной имя_переменной;



Идентификатор

Идентификатор — это имя программного объекта.

Правила задания идентификаторов:

- используются латинские буквы, цифры и знак подчеркивания (в современных стандартах допустимо использование кириллицы, но обычно не рекомендуется к использованию);
- прописные и строчные буквы различаются;
- первым символом может быть буква или знак подчеркивания, но не цифра;
- пробелы внутри имен не допускаются;
- не должен совпадать с ключевыми словами языка;
- длина идентификатора обычно ограничена 1 байтом (начиная с C++11 ограничение на длину снято)

Типы данных

точность float примерно 7 значащих цифр;

точность double примерно 16 значащих цифр;

точность long double в зависимости от того, сколько байт занимает тип данных на компьютере – от 16 до 33 значащих цифр.

Тип данных	Назначение	Размер в байтах	Диапазон значений	Пример
void	Пустой	2 или 4	-	-
int	Целочисленный	2 или 4	$-2^{15} \dots 2^{15}-1$ $-2^{31} \dots 2^{31}-1$	123
float	Числа с плавающей точкой одинарной точности	4	$3.4*10^{-38} \dots 3.4*10^{38}$	123.456
double	Числа с плавающей точкой двойной точности	8	$1.7*10^{-308} \dots 1.7*10^{308}$	1234.567891
long double	Числа с плавающей точкой расширенной точности	8, 10, 12 или 16	$3.4*10^{-4932} \dots 3.3*10^{4932}$	1.234567890123456
char	Символьный	1	0 … 255	‘Z’
bool	Логический	1	true / false	true
string	Строковый	Определяется количеством символов	-	"sadaf"

полная информация о базовых типах C++
https://en.cppreference.com/w/cpp/language/types#cite_note-1

Модификаторы порядковых типов

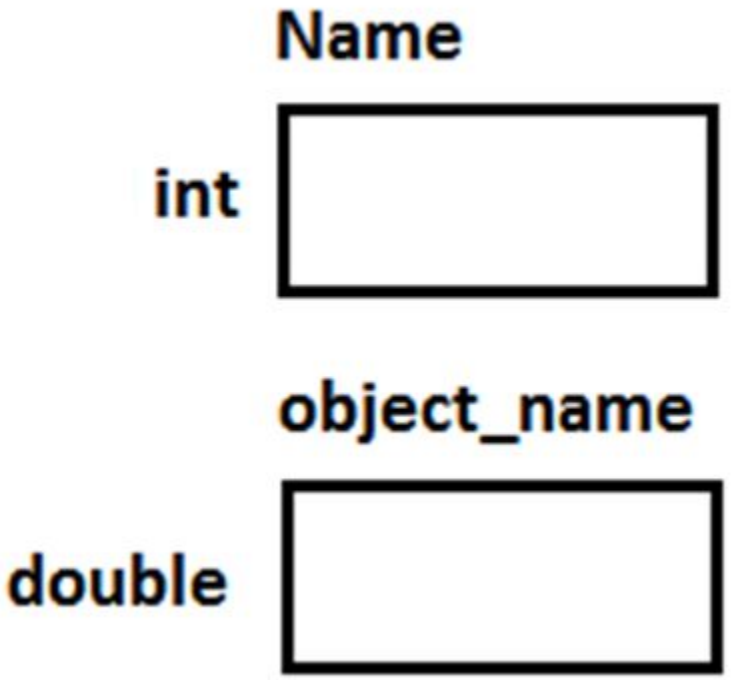
- short – короткий (только для типа int);
- long – длинный (для типов int, long int и double);
- signed – знаковый (для типов int и char);
- unsigned – беззнаковый (для типов int и char).

Влияние модификаторов на размер типа и диапазон значений

Тип	Минимальный размер в байтах	Минимальное число бит	signed	unsigned
char	1	8	$-2^7..2^7-1$	$0..2^8-1$
short int	2	16	$-2^{15}..2^{15}-1$	$0..2^{16}-1$
long int	4	32	$-2^{31}..2^{31}-1$	$0..2^{32}-1$
long long int	8	64	$-2^{63}..2^{63}-1$	$0..2^{64}-1$

Пример определения переменной

```
int Name;
double object_name;
```



Комментарии

Комментарий - это строка (или несколько строк) текста, которая вставляется в исходный код для объяснения того, что делает код, компилятор его игнорирует. Есть 2 типа комментариев: однострочные (пишется после символов //) и многострочные (между символами /* */).

```
//это однострочный комментарий
//каждый раз надо ставить символ //
//в начале строки, работает только в C++

/*а это многострочный комментарий, можно писать сколько
  угодно строчек текста, главное не забыть закрыть его,
  использовался в C, доступен в C++ */
```

Присваивание и инициализация

Оператор присваивания (=) записывает значение из правого операнда в ячейку памяти, на которую ссылается левый операнд.

Инициализация – присваивание переменной начального значения при одновременном ее определении в памяти

```
unsigned int Name = 15;  // инициализация
double object_name;      // определение в памяти
object_name = 3.5;     // присваивание
Name = Name + 3;     // присваивание
```

Оператор запятая и альтернативный способ инициализации переменной

```
int x, y; // определяем переменные x и y типа int
     // переменные x и y содержат "случайный мусор"

float z(0); // определяем и инициализируем переменную z
      // значением 0 при помощи
      // конструктора типа float
      // альтернатива float z = 0;

double d = double(); // определяем и инициализируем
     // переменную d используя конструктор по
     // умолчанию, который записывает в d значение 0
```

Цепочка операторов присваивания

```
int i, j, k; // определение переменных i, j и k

i = j = 0; // сначала в j записывается 0,
      // затем в i записывается значение j

int y = k = 10; // сначала в k записывается значение 10,
     // затем определяется в памяти переменная y
     //и в нее записывается значение k
```

Именованные константы и литералы

Значения базовых типов данных представлены литералами:

- символьные: 'g', 'к', '=';
- строковые: "Это строковая константа";
- целые: 123, 3456;
- вещественные: 1.23, 123.0, 0.1234 (тоже, что .1234);

Возможно создание именованных констант:

const тип имя = значение;

Например:

```
const int someConst = 123; //определяется именованная
   // константа someConst, проинициализированная
   // целочисленным литералом 123
```

Функция main

- должна присутствовать в любой программе на языке C++ (в приложениях с графическим интерфейсом может называться WinMain)
- с нее начинается исполнение программы

```
//глобальная область видимости
int main(){
//локальная область видимости
return 0;
}
```

Составной оператор {}

- тело функции main (как и любой другой функции) обязательно заключается в составной оператор.
- позволяет объединить несколько инструкций в одно целое, открывающаяся скобка ставится перед первой объединяемой инструкцией, закрывающаяся – после последней объединяемой;
- определяет область видимости для объектов, создаваемых внутри него;

```
int main(){//это начало внешнего составного оператора
//здесь может быть какой-то код
   {//это начало внутреннего составного оператора
   //здесь может быть какой-то код
   }//это конец внутреннего составного оператора
//здесь может быть какой-то код
}//это конец внешнего составного оператора
```

Пример области видимости программных объектов

```
int main(){
int Name;
double object_name;
//double Name; ошибка*
    {
    double Name; //ok
    }
}
```

* нельзя создавать одноименные объекты в одной области видимости

Стандартный ввод-вывод в C++

- подключить заголовочный файл iostream;
- использовать стандартное пространство имен (std);
- для вывода на экран использовать объект cout и оператор вывода <<;
- для ввода данных с клавиатуры использовать объект cin и оператор извлечения >>.

Пример вывода на экран консоли

Для одного объекта cout можно выполнить несколько каскадных операций вставки в поток (для cin – несколько каскадных извлечений из потока)
Символ "\n" внутри строкового литерала означает перенос строки
Манипулятор endl посылает команду объекту cout выполнить перенос строки и сбросить буфер потока

```
#include <iostream>
using namespace std;

int main(){
cout << "Hello, World!\n" << endl; //два переноса строки
return 0;
}
```

Пример 1 ввода-вывода в консоли

Можно выполнять вычисления непосредственно на выводе

```
#include <iostream>
using namespace std;

int main(){
cout << "Input k\n";
int k;
cin >> k;
cout << k + 1 << "\n";
system("pause");
return 0;
}
```

Пример 2 ввода-вывода в консоли

Можно не использовать using namespace std для всего кода программы, вместо этого применить std:: там, где это необходимо

```
#include <iostream>


int main(){
std::cout << "Input k\n";
int k;
std::cin >> k;
std::cout << k + 1 << std::endl;
system("pause");
return 0;
}
```

Преобразования типов

Преобразования типов обычно делят на *безопасные* (преобразования без потери данных) и *небезопасные* (с потерей данных).

Если в выражении участвуют данные разных типов, то перед вычислением все типы будут ***неявно*** преобразованы компилятором к наиболее «высокому» из типов объектов выражения, в соответствии с иерархией типов.

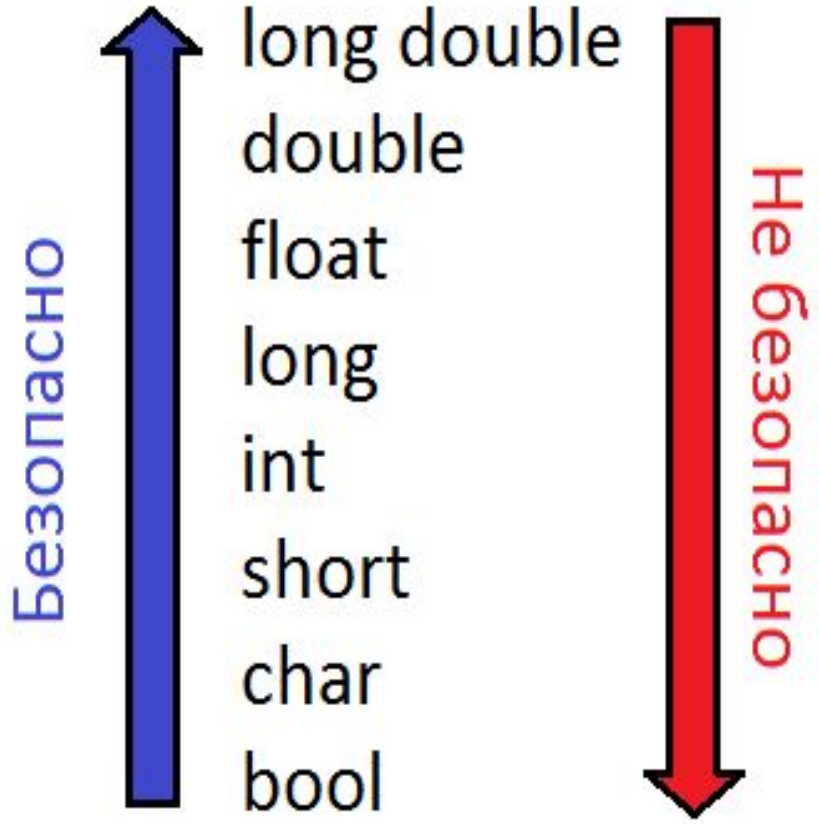
Преобразование из более «высокого» типа в более «низкий» безопасно, а из более «низкого» типа в более «высокий» - нет.

Когда программист точно знает с каким типом данных надо работать с в выражении, он использует в коде явное преобразование.

Иерархия типов данных

От высокого к низкому:

- long double
- double
- float
- long
- int
- short
- char
- bool



Примеры неявных преобразований типов

```
char c = 'x';
int i = c;      // безопасно, i = 120
int j = 'x'; // безопасно, j = 120
int a = 300;
char b = a;  // небезопасно, b = ',' – символ с кодом 44
double x = 1.5;
a = x;     // небезопасно, a = 1
```

Явное преобразование типов

- функция static_cast<новый тип>(имя) – для C++;
- операция (). Новый тип переменной записывается в круглых скобках перед ее именем или имя записывается в круглых скобках после нового типа (язык C).

```
float a, b;
cin >> a >> b;                  // например, 10.1 12.6
int c = static_cast<int>(a);  // c = 10
int d = (int) b;          // d = 12
// или так d = int (b);
```

Примеры преобразований

	Не безопасные	Безопасные

Неявные	float → int
	`float x = -1.5;`
	`int y = x; // y = -1`
		int → float
		`int x = 58;`
		`float y = x; // y = 58.0`
		`float z = y + 2; // z = 60.0`

Явные	float → int
	`float x = -1.5;`
	`int y = static_cast<int>(x);`
	`int z = (int) x;`
		int → float
		`int x = 58;`
		`float y = static_cast<float>(x);`
		`float z = (float) x;`