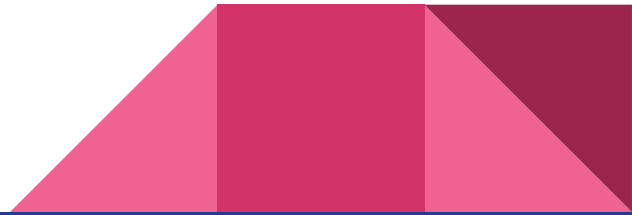
Лекция 4.

Система типов C++

Шесть основных типов

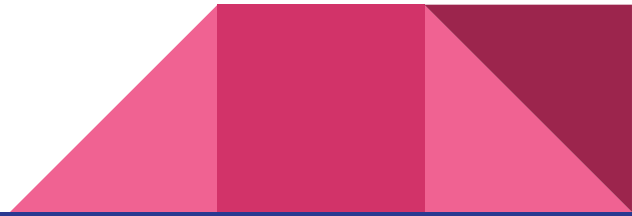
В С ++ существуют 6 элементарных типов данных:

1. Символ (**char**).
2. Целое число (**int**).
3. Число с плавающей точкой (**float**).
4. Число с плавающей точкой удвоенной точности (**double**).
5. Переменная, не имеющая значения (**void**).
6. Логический тип (**bool**).



Тип данных char

Тип данных **char** — это целочисленный тип данных, который используется для представления символов. То есть, каждому символу соответствует определённое число из диапазона [0;255]. Используя тип данных char, можно отобразить любой из 256 символов. Все закодированные символы представлены в таблице ASCII.

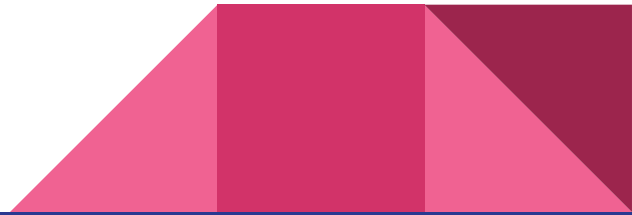


Типы данных с плавающей точкой

Типы данных **float** и **double** могут хранить как положительные, так и отрицательные числа с плавающей точкой.

У типа данных float размер занимаемой памяти в два раза меньше, чем у типа данных double.

В основном, типы данных с плавающей точкой нужны для решения задач с высокой точностью вычислений, например, операции с деньгами.



Тип данных void

Тип **void** используется для определения функции, не возвращающей никаких значений, либо для создания обобщенного указателя (generic pointer).

Тип данных bool

Тип данных **bool** — целочисленный тип данных, диапазон допустимых значений — целые числа от 0 до 255. Тип данных bool используется исключительно для хранения результатов **логических выражений**. У логического выражения может быть один из двух результатов true или false.

true — если логическое выражение истинно,

false — если логическое выражение ложно.

Константе true эквивалентны все числа от 1 до 255 включительно, тогда как константе false эквивалентно только одно целое число — 0.

Модификаторы основных типов

short — приставка укорачивает тип данных, к которому применяется, путем уменьшения размера занимаемой памяти;

long — приставка удлиняет тип данных, к которому применяется, путем увеличения размера занимаемой памяти;

unsigned (без знака) — приставка увеличивает диапазон положительных значений в два раза, при этом диапазон отрицательных значений в таком типе данных храниться не может.

Тип	Размер (байт)	Диапазон значений
bool	1	0 / 255
char	1	0 / 255
short int	2	-32 768 / 32 767
unsigned short int	2	0 / 65 535
int	2 или 4	-32 768 / 32 767 или -2 147 483 648 / 2 147 483 647
unsigned int	2 или 4	0 / 65 535 или 0 / 4 294 967 295
long int	4	-2 147 483 648 / 2 147 483 647

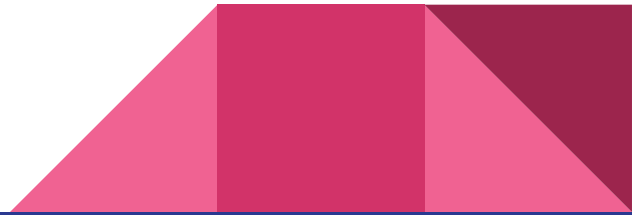
Тип	Размер (байт)	Диапазон значений
unsigned long int	4	0 / 4 294 967 295
float	4	-2 147 483 648.0 / 2 147 483 647.0
long float	8	-9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0
double	8	-9 223 372 036 854 775 808 .0 / 9 223 372 036 854 775 807.0

Совместимость типов

Существуют две разновидности совместимости типов: совместимость имен типов и совместимость структур типов.

Совместимость имен типов означает, что две переменные относятся к совместимым типам только в том случае, если они были описаны в объявлениях с одним и тем же именем типа.

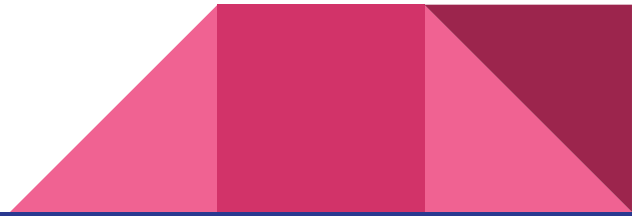
Совместимость структур типов означает, что две переменные имеют совместимые типы в том случае, если у их типов одинаковые структуры.



Структуры в C++

Язык C++ позволяет программистам создавать свои собственные **пользовательские типы данных** — типы, которые группируют несколько отдельных переменных вместе.

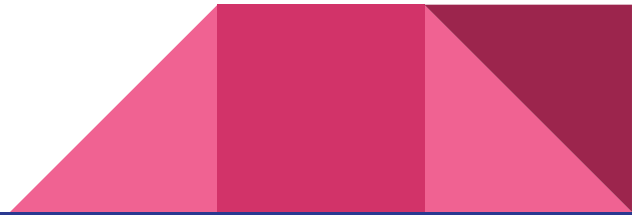
Одним из простейших пользовательских типов данных является **структура**. Структура позволяет сгруппировать переменные разных типов в единое целое.



Объявление и определение структур

Поскольку структуры определяются программистом, то вначале мы должны сообщить компилятору, как она вообще будет выглядеть. Для этого используется ключевое слово **struct**:

```
struct Employee
{
    short id;
    int age;
    double salary;
};
```

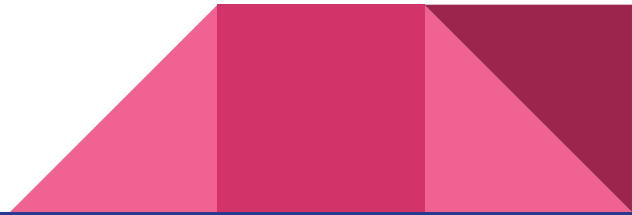


Объявление и определение структур

Чтобы использовать структуру Employee, нам нужно просто объявить переменную типа Employee:

Employee john;

Здесь мы определили переменную типа Employee с именем john. Как и в случае с обычными переменными, определение переменной, типом которой является структура, приведет к выделению памяти для этой переменной.



Доступ к членам структур

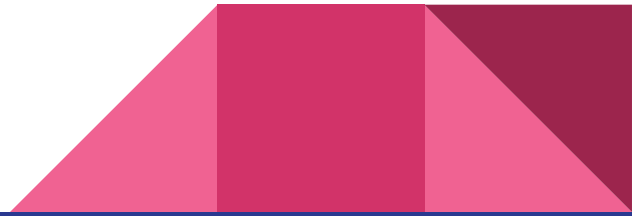
Для того, чтобы получить доступ к отдельным членам структуры, используется оператор выбора члена (.).

```
Employee john; // создаем отдельную структуру Employee для John
john.id = 8; // присваиваем значение члену id структуры john
john.age = 27; // присваиваем значение члену age структуры john
john.salary = 32.17; // присваиваем значение члену salary структуры john
```

Явное и неявное преобразование типов данных C++

В C++ различают явное и неявное преобразование типов данных. Неявное преобразование типов данных выполняет компилятор C++.

Результат любого вычисления будет преобразовываться к наиболее точному типу данных, из тех типов данных, которые участвуют в вычислении .



Неявное преобразование типов данных в C++

x	y	результат	пример
делимое	делитель	частное	x=15 y=2
int	int	int	15/2=7
int	float	float	15/2=7.5
float	int	float	15/2=7.5

Явное преобразование типов данных в C++

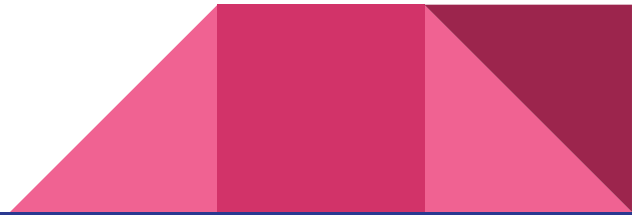
Явное преобразование данных выполняет сам программист.

Пример:

15/2=7. Результат целое число.

15.0/2=7.5 При таком делении число 15 является вещественным, значит и результат будет вещественный.

Этот же прием можно было применить к двойке, результат был бы тем же, а можно было сразу к двум числам.



Явное преобразование типов данных в C++

Еще один способ явного преобразования типов данных:

float(15) / 2 // результат равен 7.5, число 15 преобразуется в вещественный тип данных float.

double(15) / 2 // результат равен 7.5

В C++ также предусмотрена унарная операция приведения типа:

static_cast</*тип данных*/>(/*переменная или число*/)

Пример: **int ret=15;**

static_cast<float>(ret)/2 //результат равен 7.5

Преобразование строки в число в C ++

Функции **atoi** (для целых чисел (от алфавита до целого числа)) и **atof** (для значений с плавающей запятой (от алфавита до плавающей)).

```
#include <cstdlib> //the standard C library header

#include <string>

int main()

{    std::string si = "12";

    std::string sf = "1.2";

    int i = atoi(si.c_str()); //the c_str() function "converts"

    double f = atof(sf.c_str()); //std::string to const char*  }
```