**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №1**

**по дисциплине «Программирование»**

Тема: “Типы данных и их внутреннее представление в памяти”

|  |  |  |
| --- | --- | --- |
| Студент гр. 2372 |  | Васильев Ю.А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2022

**Цель работы.**

Изучение внутреннего строения памяти, понимания, как хранятся типы данных.

**Основные теоретические положения.**

Память состоит из непрерывной последовательности байтов (8 бит). Каждый байт памяти имеет адрес, используя который можно записать данные в байт или прочитать их.

Но простой доступ к памяти может создавать коллизии, т.к. нет определённых границ между байтами, т.е. изменение первой переменной может привести к изменению второй.

Чтобы решить эту проблему, в языках программирования были созданы типы данных.

Каждый тип данных определяет:

* Характер данных (как обращаться с байтами в памяти)
* Объём памяти (сколько байтов отведено под тип)
* Множество значений (какие значения может принимать тип)
* Возможные операции с типом

Например, в языке С существуют следующие основные типы данных:

* int (целый)
* float (вещественный)
* double (вещественный, увеличенная точность)
* bool (логический)
* char (символьный)

Так же существуют модификаторы типов данных, которые влияют на их хранение в памяти (применяется к стандартным типам):

* short (короткий)
* long (длинный)
* signed (знаковый)
* unsigned (беззнаковый)

В языке С переменную можно определить так:

<тип данных> <имя> [инициализатор]

Например, конструкция определения переменной var1 с типом int и значением 42:

int var1 = 42;

Вся эта конструкция является инструкцией языка программирования, поэтому после нее всегда должен ставиться символ ‘;’.

Разберём как хранится целочисленный тип данных в памяти (для 32-битного процессора):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 32 | 31 | 30 | 29 | … | 5 | 4 | 3 | 2 | 1 |
| 0 | 1 | 1 | 0 |  | 1 | 0 | 1 | 1 | 0 |

* 32й бит отвечает за знак (если signed)
* Остальные биты представляют значение числа в двоичной системе счисления.

Если целое число используется с модификатором unsigned, то знаковый бит используется для представления числа в двоичной системе счисления.

Теперь разберём как хранятся вещественные типы данных в памяти:

Каждое вещественное число можно представить в виде мантиссы, умноженной на экспоненту (экспоненциальный формат). Например:

**3.14** = **0.314**e**+1**

Любое вещественное число в памяти представляется так:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 32 | 31 | 30 | 29 | … | 5 | 4 | 3 | 2 | 1 |
| 0 | 1 | 0 | 0 |  | 1 | 1 | 0 | 0 | 1 |

* 32й бит отвечает за знак
* 31-24 биты отвечают за экспоненту числа
* Остальные биты отвечают за мантиссу числа

**Постановка задачи.**

Необходимо вывести на экран количество байтов, занимаемых типами и двоичное представление чисел типов данных int, float, double.

1. Вывод количества занимаемых типом байтов с помощью оператора sizeof.

Вывод двоичного представления int:

1.   Создание маски из единицы, её сдвиг на знаковый бит.

2.   Битовая операция “И” с самим числом и маской.

3. Приведение результата битовой операции к типу bool

4. Вывод на экран 1 или 0, в зависимости от значения bool.

5. Побитовый сдвиг маски вправо на 1.

6. Повторение шагов 1-4, пока не будут выведены все биты числа.

Вывод двоичного представления float:

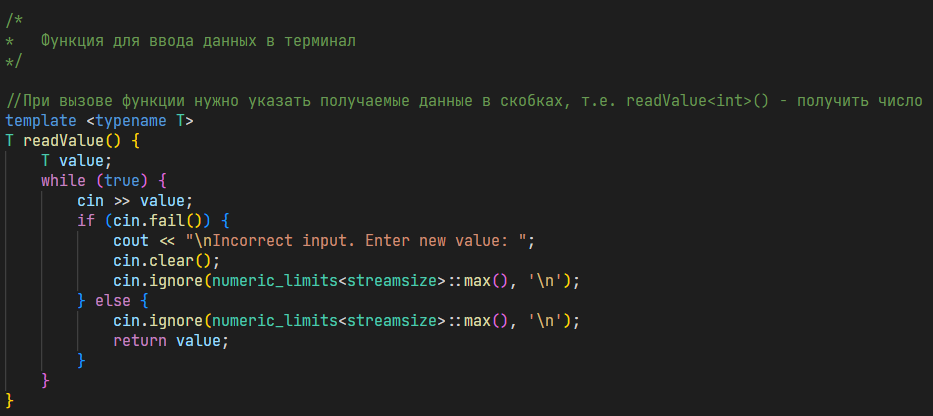
1. Использование union для объединения участка памяти float и int (нужно, чтобы совершать побитовые операции с числом)
2. Повторение шагов для int.

Вывод двоичного представления double:

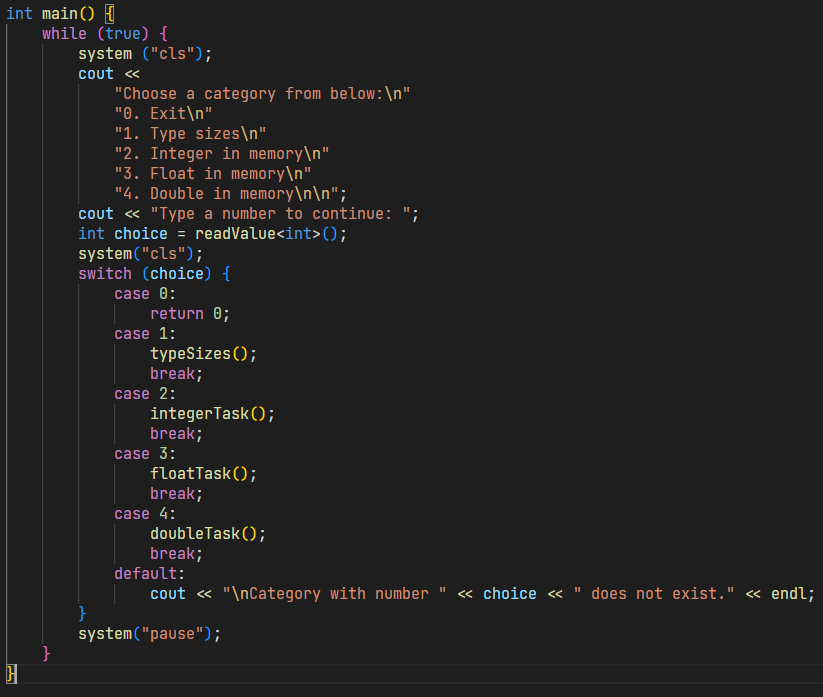
1. Использование union для объединения участка памяти float и массива int с двумя элементами (нужно, чтобы совершать побитовые операции с числом)
2. Повторение шагов для каждого элемента int в массиве.

**Выполнение работы.**

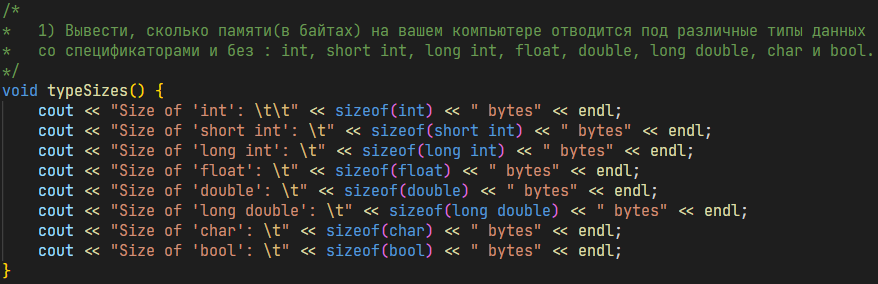
Вспомогательная функция ввода в терминал и получения из него данных в переменную (принимает только указанный тип, иначе выводит ошибку):



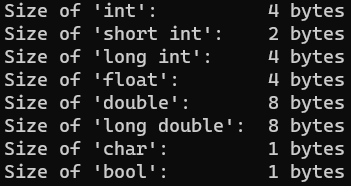
Основная функция, меню выбора функционала программы. Для перехода нужно указать число 0-4.



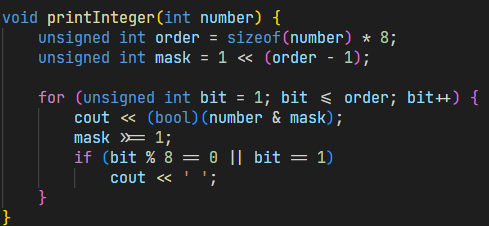
Задача 1:

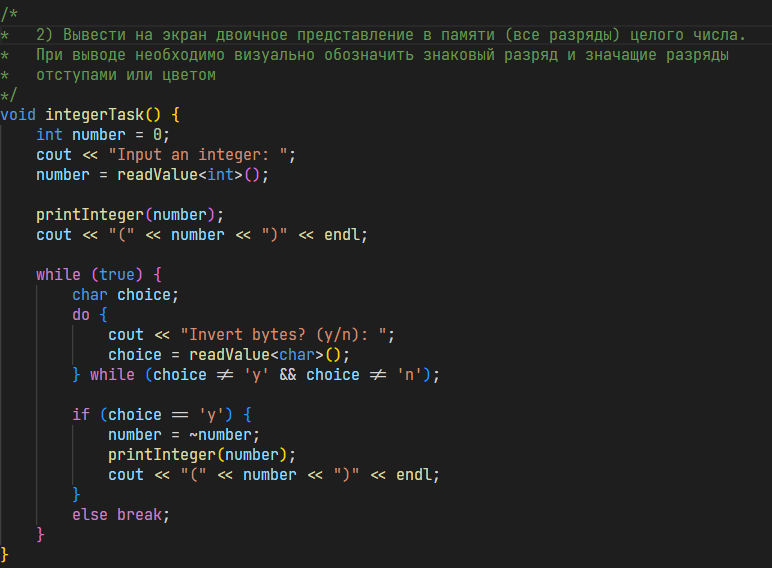


Вывод задачи:

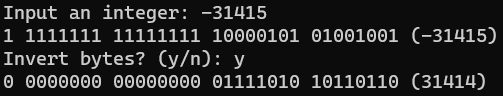


Задача 2:

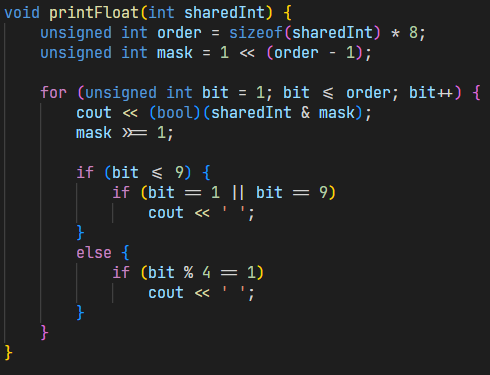


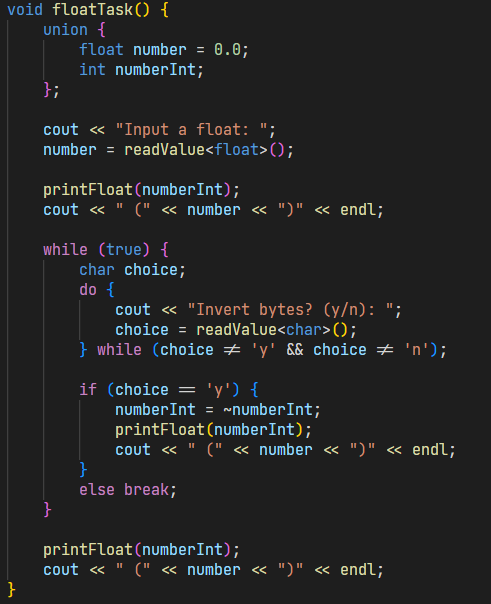


Вывод задачи для числа -31415:

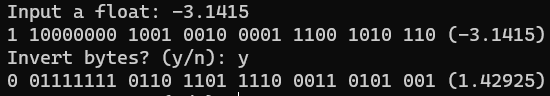


Задача 3:

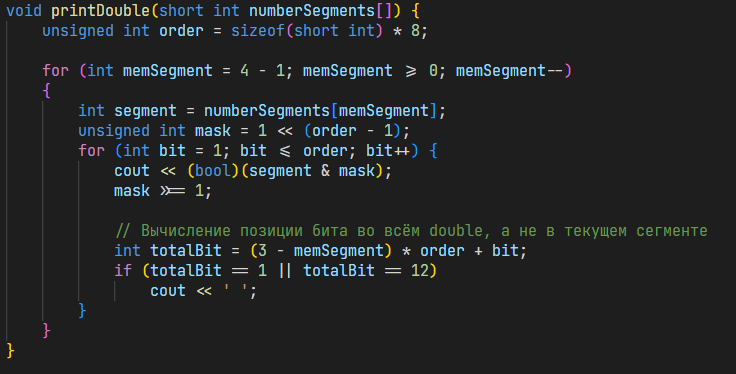


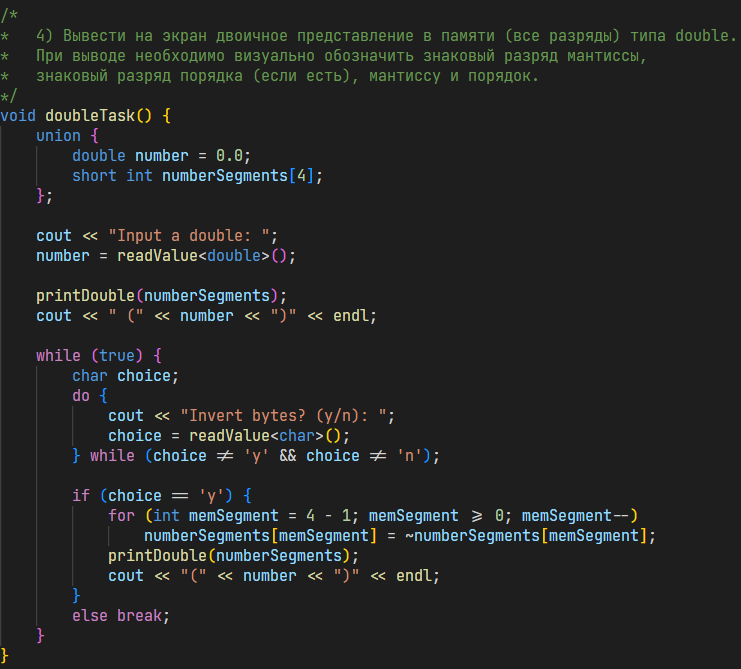


Вывод задачи для числа -3.1415:

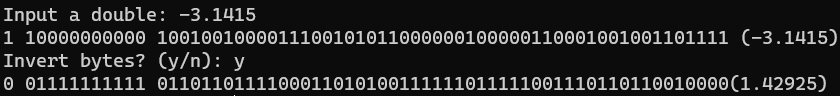


Задача 4:





Вывод задачи для числа -3.1415:



**Выводы.**

Я научился выводить двоичное представление типов данных (значения битов, которые хранятся в памяти), изучил внутреннее строение целых и вещественных чисел.