**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(ка) гр. |  | Дементьев М. А. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2021

**Цель работы.**

Изучение и организация принципов работы с памятью; получение практических навыков в работе с памятью.

**Основные теоретические положения.**

**Директива include** является директивой препроцессора. Каждая директива препроцессора размещается на одной строке. И в отличие от обычных инструкциий языка C++, которые завершаются точкой с запятой ; , признаком завершения препроцессорной директивы является перевод на новую строку. Кроме того, директива должна начинаться со знака решетки #. Непосредственно директива "include" определяет, какие файлы и библиотеки надо подключить в данном месте в код программы.

#include <iostream>

Каждая программа на языке С++ должна иметь как минимум одну функцию - **функцию main()**. Именно с этой функции начинается выполнение приложения. Ее имя main фиксировано и для всех программ на Си всегда одинаково.

Функция также является блоком кода, поэтому ее тело обрамляется фигурными скобками, между которыми определяется набор инструкций.

Определение функции main начинается с возвращаемого типа. Функция main в любом случае должна возвращать число. Поэтому ее определение начинается с ключевого слова int.

В частности, при создании первой программы использовалась следующая функция main:

int main{

…

}

**Комментарии** – это фрагменты текста, игнорирующиеся компилятором при обработке текста программы. Комментарии в текстах программ используются для различных пояснений к тексту программы, а также для исключения временно не нужных фрагментов текста программы (например, отладочных кодов или вариантов реализации).

В C++ имеются комментарии двух видов: однострочные и многострочные.

Однострочный комментарий начинается двумя символами // (прямой косой черты) и заканчивается в конце строки текста программы.

// Однострочный комментарий

**Объект std::cout** (который находится в библиотеке iostream) используется для вывода данных на экран (в консольное окно). Для вывода нескольких предложений на одной строке оператор вывода << нужно использовать несколько раз, например:

std::cout << a << b;

**Объект std::cin** является противоположностью std::cout. В то время как std::cout выводит данные в консоль с помощью оператора вывода <<, **std::cin** получает данные от пользователя с помощью **оператора ввода** >>. Используя std::cin мы можем получать и обрабатывать пользовательский ввод:

std::cin >> a;

**Условный оператор if** - самый простой оператор для принятия решения. Он используется для определения того, будет ли выполняться определенный оператор или блок операторов, т.е. если определенное условие истинно, тогда блок оператора выполняется иначе нет.

if(условие)

{

// выполнение кода если

// условие истинно (true)

}

**Конструкция for** используется для повторения блока операторов, заключенных в фигурные скобки. Счетчик приращений обычно используется для приращения и завершения цикла. Оператор **for** подходит для любых повторяющихся действий и часто используется в сочетании с массивами коллекций данных/выводов.

Заголовок цикла for состоит из трех частей:

for ( **initialization**; **condition**; **increment**) {

//операторы выполняющиеся в цикле

}

Инициализация (**Initialization**) выполняется самой первой и один раз. Каждый раз в цикле проверяется условие (**condition**), если оно верно, выполняется блок операторов и приращение (**increment**), затем условие проверяется вновь. Когда логическое значение условия становится ложным, цикл завершается.

**Объединения** – это две или более переменных расположенных по одному адресу (они разделяют одну и ту же память). Объединения определяются с использованием ключевого слова **union**. Объединения не могут хранить одновременно несколько различных значений, они позволяют интерпретировать несколькими различными способами содержимое одной и той же области памяти.

union{

int a;

float b;

}

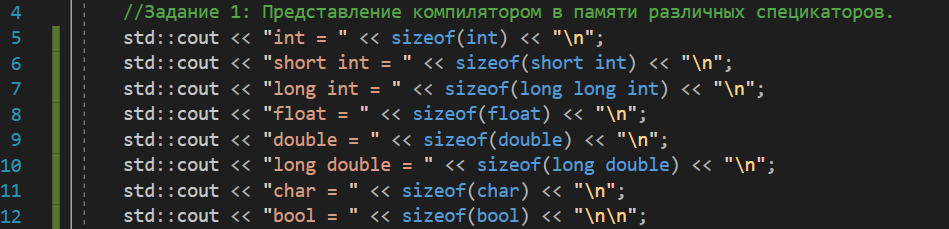
**Постановка задачи.**

Разработать алгоритм и написать программу, которая позволяет:

1. Вывести, сколько памяти (в байтах) на вашем компьютере отводится под различные типы данных со спецификаторами и без: int, short int, long int, float, double, long double, char и bool.
2. Вывести на экран двоичное представление в памяти (все разряды) целого числа. При выводе необходимо визуально обозначить знаковый разряд и значащие разряды отступами или цветом.
3. Вывести на экран двоичное представление в памяти (все разряды) типа float. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

**Выполнение работы.**

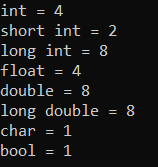
Задание 1:



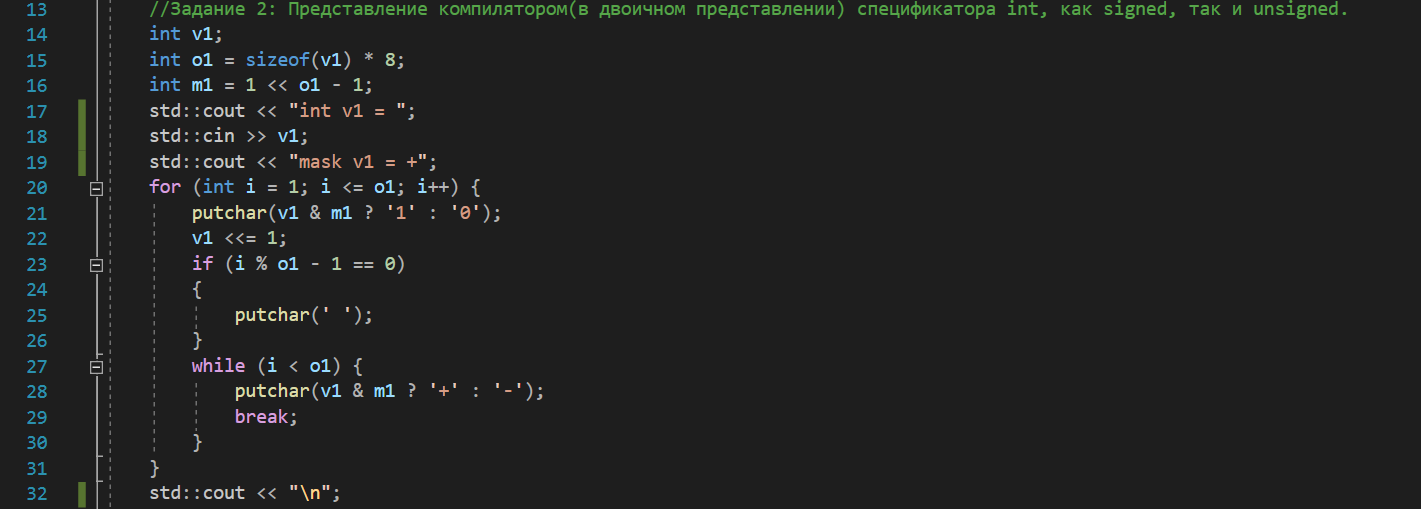
Блок описания кода и использованных алгоритмов

Задействовав оператор **sizeof** мы получаем побайтовое представление, поставленных нами в задаче, спецификаторов в памяти компилятора.

Блок скриншотов работы программы:



Задание 2:



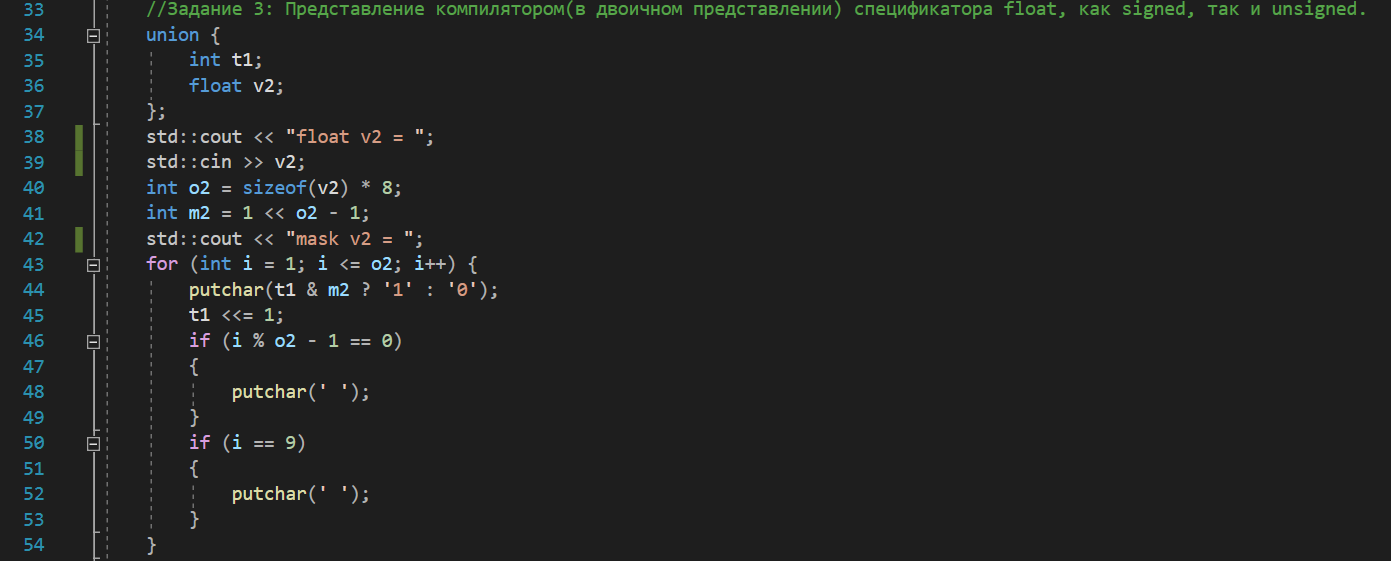
Блок описания кода и использованных алгоритмов

Цикл for представляет собой “счётчик” для идентификации разрядов(под средством переменной **i**) , с которыми будут производится операции, путём побитового представления **v1** и сравнения значения **i** с **o1**, где **o1** – это переменная отвечающая за кол-во побитовых разрядов в поставленном нами типе данных. Под средством использования функции putchar мы, при помощи тернарной операции, производим побитовое сравнение **v1** и **m1**, где **m1** - маска побитового сравнения, соответствующая поставленному в задаче типу данных, путём побитового сдвига при помощи **<<** влево. Результатом работы является значение **1**, при истине, и **0**, при её отсутствии. Каждый раз, при проходе по циклу, переменная **v1** сдвигается влево для сравнения с первым битом маски **m1**. Для отделения знакового разряда от разрядов цифровой части числа, двоичный код его абсолютной величины, используем функцию **putchar** и **if** для вывода пробела, и определения знакового разряда, соответственно. Путём использования цикла **while** и функции **putchar** мы обозначаем значащие и незначащие разряды знаками “+” и “-“ соответственно.

Блок скриншотов работы программы:



Задание 3:



Блок описания кода и использованных алгоритмов

Алгоритм никак не отличается от задания 2, за исключением использования объединения. Которое позволяет производить операции с вещественными числами, путём объединения области памяти с числом, с которым можно использовать эти операции.

Блок скриншотов работы программы:



**Выводы**

Мы изучили организацию принципов работы с памятью и получили практические навыки в работе с памятью.