**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. |  | Мельников В.С. |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**Цель работы.**

Изучение и организация типов данных и их представление в памяти; получение практических навыков работы с типами данных; определение преимуществ и недостатков использования различных типов данных.

**Основные теоретические положения.**

Основные (предопределенные) типы данных часто называют арифметическими, поскольку их можно использовать в арифметических операциях. Для описания основных типов определены следующие ключевые слова:

* ·         **int** (целый);
* ·         **float** (вещественный);
* ·         **double** (вещественный тип с двойной точностью);
* ·         **bool** (логический);
* ·         **char** (символьный).
* Типы **int**,**bool**и**char**  относят к группе целочисленных (целых) типов, а **float**и**double** - к группе вещественных типов - типов с плавающей точкой. Код, который формирует компилятор для обработки целых величин, отличается от кода для величин с плавающей точкой.
* Существует четыре спецификатора типа, уточняющих внутреннее представление и диапазон значений стандартных типов:
* ·         **short**(короткий);
* ·         **long** (длинный);
* ·         **signed** (знаковый);
* ·         **unsigned** (без знаковый).

Для определения размера отведенного под определенный тип данных, надо использовать функцию **sizeof().**

Величины логического типа могут принимать только значения **true** и **false**, являющиеся зарезервированными словами.

Внутренняя форма представления значения **false** - О (нуль). Любое другое значение интерпретируется как **true**. При преобразовании к целому типу **true** имеет значение **1**(единица).

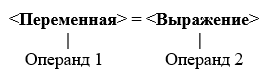
* **Поразрядные логические операции**
* К этой группе операций относятся:
* ·         **~**    - побитовое отрицание (побитовое НЕ) - унарная операция;
* ·         **&**   - побитовая конъюнкция (побитовое И) - бинарная операция;
* ·         **|**     - побитовая дизъюнкция (побитовое ИЛИ) – бинарная операция;
* ·         **^**    - побитовое исключающее ИЛИ - бинарная операция.

Эти операции необходимы для того, чтобы работать

представлением типов данных в памяти.

Также в работе использованы Циклы **for** и **while,** операции присваивания и логические операции.

### Операция присваивания



### Логические операции

* ·         **!**     - логическое отрицание (логическое НЕ);
* ·         **&&** - конъюнкция (логическое И);
* ·         **||**       - дизъюнкция (логическое ИЛИ).

**Постановка задачи.**

Запишем, сколько места необходимо под типы переменных int, short int, long int, float, double, long double, char и bool. Также предстоит написать программы для представления (int, float, double) в памяти и для смены чётных битов на 0, а нечетных на 1 в битовом представлении.

Алгоритм работы 1 функции:

1. Пользователь вводит число
2. Показывается представление числа в памяти
3. С помощью пробелов выделяются байты, знаковый бит. С помощью цветов выделяется знаковый бит, мантиса, экспонента.

Алгоритм работы 2 функции:

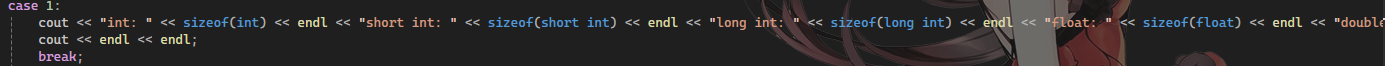
1)Если число отрицательное, то все биты, стоящие на чётных позициях меняются на 0 благодаря побитовому и(&) и отрицанию, если число положительное, то все биты, стоящие на нечётных позициях меняются на 1 благодаря побитовому или(|).

2)Представление в памяти выводится в консоль.

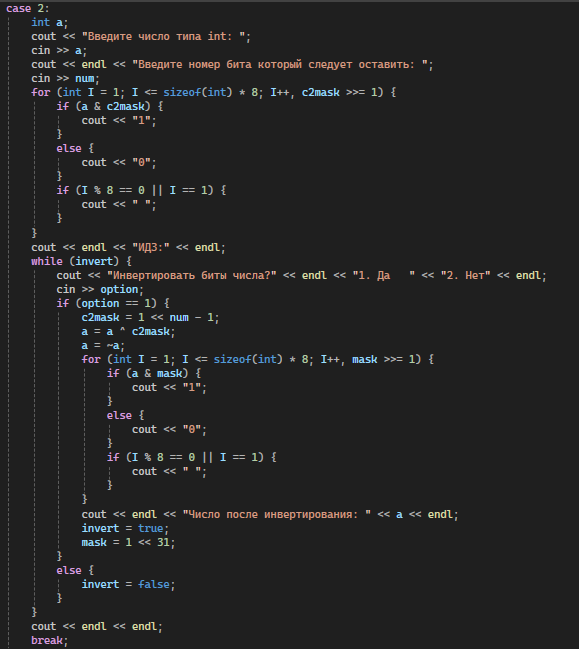
**Выполнение работы.**

**1 программа:**

1) Сколько памяти занимает каждый тип переменной



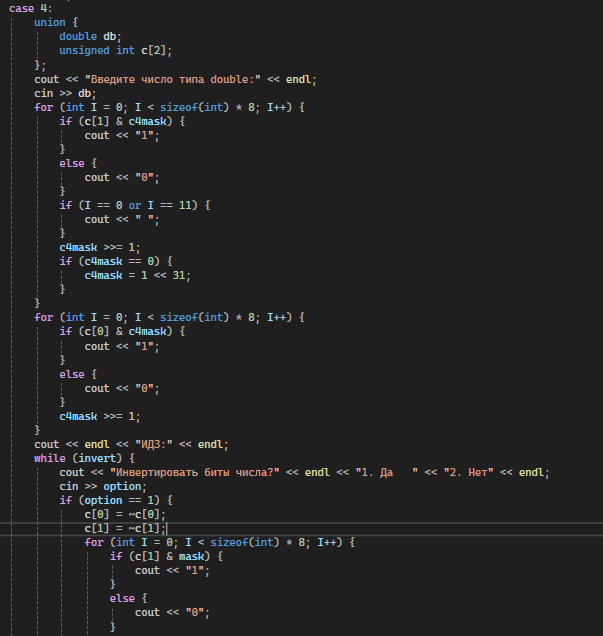
2) Вводим число типа int и видим его побитовое представление в консоли. Также пользователь может ввести на сколько битов произвести сдвиг влево, после этого программа покажет преобразованное число.

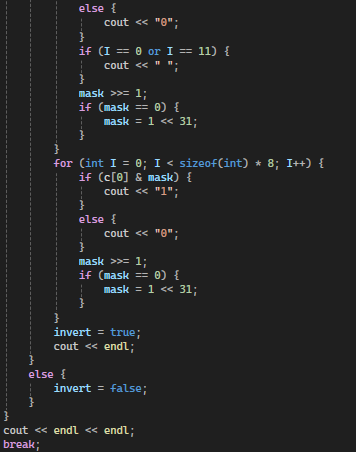


3) Вводим число типа float и видим его побитовое представление в консоли. Также пользователь может ввести на сколько битов произвести сдвиг влево, после этого программа покажет преобразованное число.



4) Вводим число типа double и видим его побитовое представление в консоли. Также пользователь может ввести на сколько битов произвести сдвиг влево, после этого программа покажет преобразованное число.





**Вывод.**

Были получены знания о представлении разных типов данных в памяти.

Научились использовать побитовые операции.