**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

|  |  |  |
| --- | --- | --- |
| Студентка гр. 4372 | Кубрина Анастасия |  |
| Преподаватель |  | Глущенко А. Г. |

Санкт-Петербург

2024

**Цель работы.**

Изучение типов данных и их внутреннее представление в памяти.

**Основные теоретические положения.**

Типы данных играют ключевую роль в любом языке программирования, включая C++. Они определяют, какие значения могут храниться в переменных, а также операции, которые могут быть выполнены с этими значениями. Понимание внутреннего представления типов данных в памяти позволяет разработчикам оптимизировать производительность и использовать ресурсы более эффективно. В этом отчете мы рассмотрим основные типы данных в C++, их характеристики и способы хранения в памяти.

1. Основные типы данных в C++

1.1. Примитивные типы данных

C++ поддерживает несколько примитивных типов данных:

• Целочисленные типы (int): Хранят целые числа. В зависимости от платформы могут занимать 2, 4 или 8 байт (например, int, short, long, long long).

• Типы с плавающей точкой (float, double): Используются для хранения чисел с дробной частью. float обычно занимает 4 байта, а double — 8 байт.

• Логические типы (bool): Хранят значения истинности (true/false). Обычно занимают 1 байт, хотя фактически могут занимать больше из-за выравнивания.

• Символьные типы (char): Хранят отдельные символы. char занимает 1 байт и может представлять символы в кодировке ASCII.

2. Внутреннее представление типов данных в памяти

2.1. Алгоритм хранения

Каждый тип данных имеет свое внутреннее представление в памяти, которое определяется его размером и форматом:

• Целочисленные типы: Хранятся в двоичном формате. Например, 32-битное целое число (int) занимает 4 байта.

• Числа с плавающей точкой: Хранятся в формате IEEE 754, где число представляется как мантисса и экспонента.

• Строки: Строки в C++ хранятся как массив символов с добавленным нулевым терминатором ('0') для обозначения конца строки.

**Постановка задачи.**

Необходимо выполнить четыре задания:

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.

4) Вывести на экран двоичное представление в памяти (все разряды) типа double. При выводе необходимо визуально обозначить знаковый разряд мантиссы, знаковый разряд порядка (если есть), мантиссу и порядок.

**Выполнение работы.**

Код программы представлен в приложении А.

Блок описания кода и использованных алгоритмов:

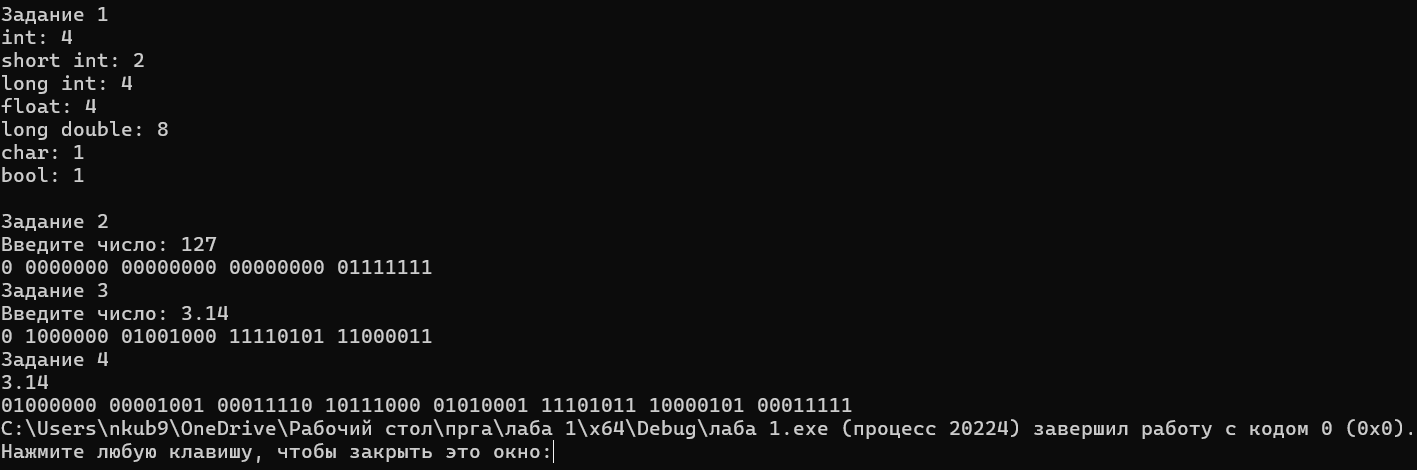
В первом задание, мы, с помощью sizeof(), выводим сколько памяти на моем компьютере отводится под различные типы данных.

Во-втором задание, мы выводим на экран двоичное представление в памяти целого числа, которое задаст пользователь.

В-третьем, мы выводим на экран двоичное представление в памяти типа float.

В четвертом, мы выводим на экран двоичное представление в памяти типа double.

Блок скриншотов работы программы



Блок таблицы с тестовыми данными:

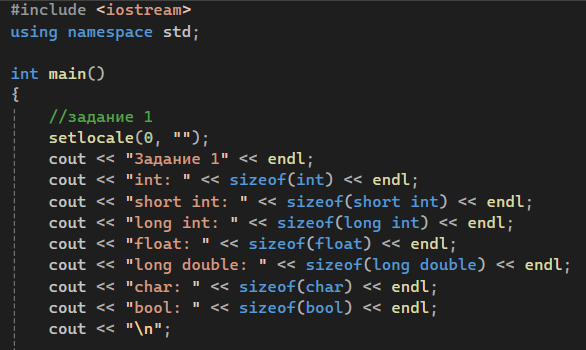
|  |  |
| --- | --- |
| Short int | 2 байт |
| Long int | 4 байт |
| Float | 4 байт |
| Long double | 8 байт |
| Char | 1 байт |
| Bool | 1 байт |
| 127 | 0 0000000 00000000 00000000 01111111 |
| 3.14 | 0 1000000 01001000 11110101 11000011 |
| 3.14 | 01000000 00001001 00011110 10111000 01010001 11101011 10000101 00011111 |

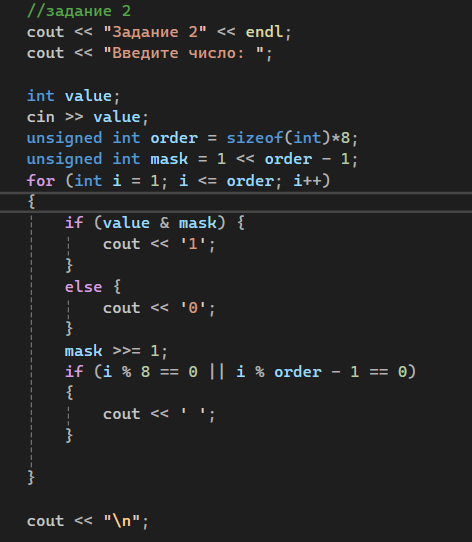
**Выводы.**

Типы данных и их внутреннее представление в памяти являются важными аспектами разработки программного обеспечения на C++. Знание о том, как данные хранятся и управляются в памяти, помогает разработчикам создавать более эффективные и производительные приложения. Понимание особенностей различных типов данных позволяет лучше использовать возможности языка C++ и оптимизировать код.

Приложение А

рабочий код





Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание