**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. | Санько Е.А. |  |
| Преподаватель | Глущенко А. Г. |  |

Санкт-Петербург

2022

**Цель работы.**

Изучить внутреннее представление памяти. Получить практические навыки работы со встроенными типами данными и объединениями.

**Основные теоретические положения.**

В зависимости от используемого компилятора встроенные типы данных могут занимать разное количество байт в памяти. Чтобы узнать сколько байт весит тот или иной тип данных можно использовать встроенную функцию sizeof(type\_name), которая возвращает размер типа. Для побитовых операций есть специальные операторы (>>, <<, |, &, ^), которые позволяют делать некие поразрядные операции. Ключевое слово union позволяет создавать объединения объектов, после которого они будут занимать одну область в памяти.

**Постановка задачи.**

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.**

Полный код программы:

#include <iostream>

using namespace std;

int main()

{

cout << "\t\tTask 1\n";

cout << "int: " << sizeof(int) << "\n";

cout << "short int: " << sizeof(short int) << "\n";

cout << "long int: " << sizeof(long int) << "\n";

cout << "float: " << sizeof(float) << "\n";

cout << "double: " << sizeof(double) << "\n";

cout << "long double: " << sizeof(long double) << "\n";

cout << "char: " << sizeof(char) << "\n";

cout << "bool: " << sizeof(bool);

cout << "\n\n\t\tTask 2\n";

cout << "Enter int: ";

int input;

cin >> input;

int a = sizeof(int) \* 8;

int mask = 1 << a - 1;

for (int i = 1; i <= a; i++)

{

cout << ((input & mask) ? 1 : 0);

if (i == 1 || i % 8 == 0)

cout << " ";

input <<= 1;

}

cout << "\n\n\t\tTask 3\n";

union {

int tool;

float f;

};

cout << "Enter float: ";

cin >> f;

for (int i = 1; i <= a; i++)

{

cout << ((tool & mask) ? 1 : 0);

if (i == 1 || i % 8 == 0)

cout << " ";

tool <<= 1;

}

cout << "\n\n\t\tTask 4\n";

union {

double d;

int arr[2];

};

cout << "Enter double: ";

cin >> d;

for (int i = 1; i <= a; i++)

{

cout << ((arr[1] & mask) ? 1 : 0);

if (i == 1 || i % 8 == 0)

cout << " ";

arr[1] <<= 1;

}

for (int i = 1; i <= a; i++)

{

cout << ((arr[0] & mask) ? 1 : 0);

if (i % 8 == 0)

cout << " ";

arr[0] <<= 1;

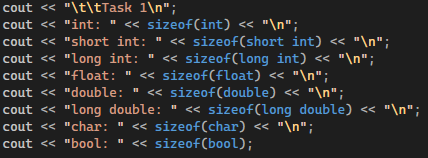
}

cout << "\n\n";

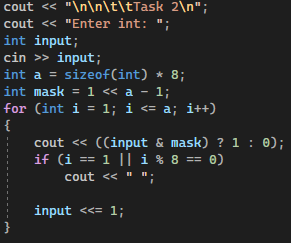
}

Алгоритм выполнения программы:

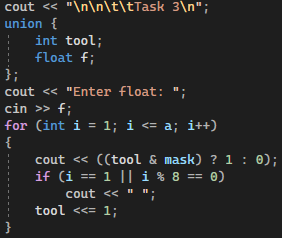
1. Используем операцию sizeof(), определяющую размер типа данных.



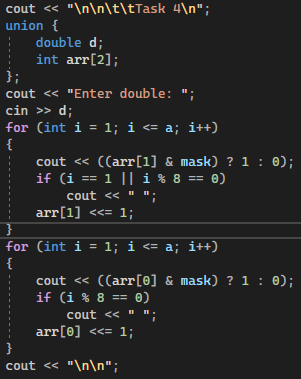
1. Вводим с клавиатуры целое число. С помощью побитового сдвига создаём маску размерностью 32 бит (совпадающей размерностью с типом int) вида 100..00 (1 единица и 31 ноль). Циклом for 32 раза проверяем первый разряд исходного числа с помощью условной операции (&:) и побитового и (&), в конце каждой итерации делаем сдвиг влево на 1 разряд. Получаем двоичный код исходного целого числа, первый бит которого отводится под знак.



1. Поскольку вещественные типы данных хранятся в памяти компьютера иначе, чем целочисленные, создаем объединение из переменной типа float и переменной типа int. Вводим с клавиатуры вещественное число. Повторяем описанный в пункте 2 алгоритм со вторым (целым числом). Получаем двоичный код исходного вещественного числа, первый бит которого отводится под знак, следующие 8 под экспоненту, оставшиеся 23 под мантиссу.



1. Для применения вышеописанного алгоритма для типа double также создадим объединение. На этот раз из переменной типа double и массива из двух целочисленных элементов (поскольку под тип double отводится в 2 раза больше байт, чем под тип int). Далее действуем по алгоритму, описанному в пункте 3. Получаем двоичный код исходного числа, 1 бит которого отводится под знак, следующие 11 под экспоненту и оставшиеся 52 под мантиссу.



**Выводы.**

Я изучил внутреннее представление данных. Научился определять размерности типов данных, выводить двоичные представления переменных типа int, float и double, благодаря использованию побитовых операций. Функция объединения union позволяет хранить две или более переменных, расположенных по одному адресу, что помогает при выводе двоичного кода числа с плавающей точкой (функция union разделяет число на мантиссу и порядок). Для вывода двоичной записи типа double необходимо привести мантиссу и порядок к одной размерности, в чем помогла работа с массивом.