**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. | Цатинян А.А. |  |
| Преподаватель | Глущенко А. Г. |  |

Санкт-Петербург

2022

**Цель работы.**

Изучение внутреннего представления памяти. Получение практических навыков работы с 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.**

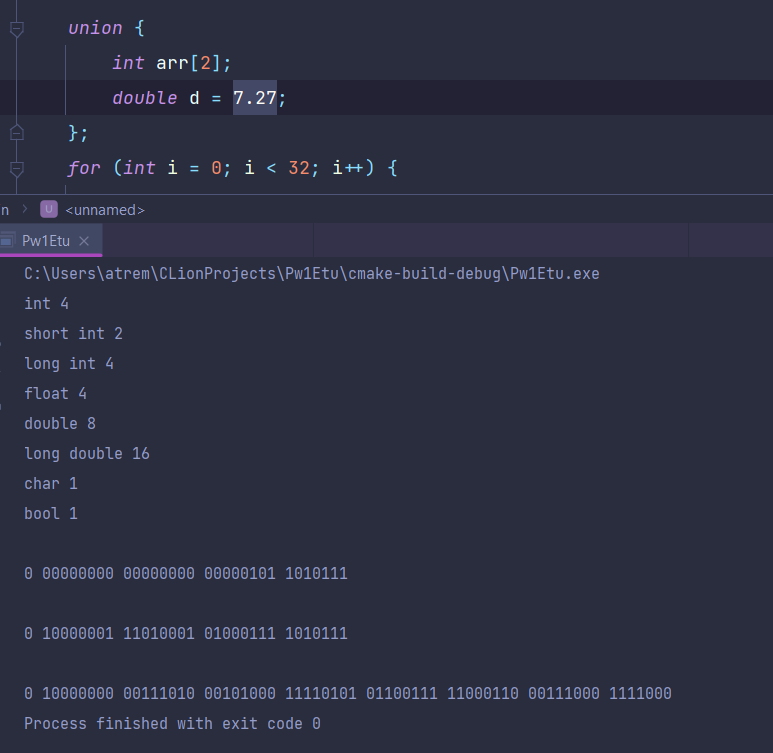
Код программы.

#include <iostream>  
  
*using namespace* std;  
  
*void* binaryRepresentation(*int* &x) {  
 *int* m = 1 << 31;  
 *for* (*int* i = 0; i < 32; i++) {  
 cout << ((x & m) ? 1 : 0);  
 *if* (i % 8 == 0) cout << " ";  
 x <<= 1;  
 }  
}  
  
*int* main() {  
 cout << "int " << *sizeof*(*int*) << endl << "short int " << *sizeof*(*short int*) << endl << "long int " << *sizeof*(*long int*)  
 << endl << "float " << *sizeof*(*float*) << endl << "double " << *sizeof*(*double*) << endl << "long double " << *sizeof*(*long double*)  
 << endl << "char " << *sizeof*(*char*) << endl << "bool " << *sizeof*(*bool*) << endl << endl;  
  
 *int* x;  
 cout << "Enter an integer" << endl;  
 cin >> x;  
 binaryRepresentation(x);  
 cout << "\n\n";  
  
 *union* {  
 *int* fa;  
 *float* f;  
 };  
 cout << "Enter float " << endl;  
 cin >> f;  
 binaryRepresentation(fa);  
 cout << "\n\n";  
  
 *union* {  
 *int* arr[2];  
 *double* d;  
 };  
 cout << "Enter double " << endl;  
 cin >> d;  
 binaryRepresentation(arr[0]);  
 binaryRepresentation(arr[1]);  
}

Блок описания кода и использованных алгоритмов

1. Используя функцию sizeof() выводим размеры требуемых типов подряд
2. Тип int весит 32 бита. Создаём другую переменную такого же размера. Использую побитовый сдвиг приводим её в вид 100…00 в двоичном представлении. И в цикле 32 раза проверяем первый разряд исходного числа с помощью побитового и(&), а потом делаем сдвиг, чтобы проверить следующий разряд.
3. У типов с плавающей точкой не перегружены операторы побитовых операций, поэтому создаём объединение этой переменной с пустой переменной целого типа, того же размера. Теперь можно выполнить алгоритм описанные в пункте 2, с целочисленной переменной из объединения и все эти операции будут отражаться на значении переменной с плавающей точкой

Блок скриншотов работы программы



**Выводы.**

Побитовые операции позволяют исследовать и работать с двоичным представлением переменных. Объединения позволяют работать со значениями переменных, с помощью операторов, которые у этих типов не перегружены.