**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по лабораторной работе №1**

**по дисциплине «Организация ЭВМ и Систем»**

Тема: Исследование внутреннего представления различных форматов данных

|  |  |  |
| --- | --- | --- |
| Студент(ка) гр. |  | Полуянов В. Н. |
| Преподаватель |  | Гречухин М. Н. |

Санкт-Петербург

2023

**Цель работы.**

Знакомство с внутренним представлением различных типов данных, используемых компьютером при их обработке.

**Основные теоретические положения.**

Представление положительных и отрицательных числе в прямом, обратном и дополнительном кодах отличается. В прямом коде в знаковый разряд помещается цифра 1, а в разряды цифровой части числа – двоичный код его абсолютной величины.

Обратный код получается инвертированием всех цифр двоичного кода абсолютной величины, включая разряд знака. Прямой код можно преобразовать в обратный, инвертировав все значения битов.

Дополнительный код получается образованием обратного кода с прибавлением единицы к его младшему разряду.

Увидеть, как тип данных представляется на компьютере, можно при помощи побитового сдвига и поразрядной конъюнкции.

При сдвиге вправо для положительных чисел освобожденные позиции битов, заполняются нулями. Для отрицательных – единицами. Сдвиг влево является логическим сдвигом (биты, сдвигаемые с конца, отбрасываются, включая бит знака).

Внутреннее представление вещественного числа состоит из двух частей – мантиссы и порядка (экспоненты). Для 32-разрядного процессора для float под мантиссу отводится 23 бита, под экспоненту – 8, под знак – 1. Для double под мантиссу отводится 52 бита, под экспоненту – 11, под знак – 1.

**Постановка задачи.**

Разработать алгоритм и написать программу, которая позволяет:

1. Вывести, сколько памяти (в байтах) отводится под различные типы данных со спецификатором и без: int, short int, long int, float, double, long double, char, bool.
2. Вывести на экран двоичное представление в памяти (все разряды) целого числа. При выводе необходимо визуально обозначить знаковый разряд и значащие разряды.
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 ест), мантиссу и порядок.

**Выполнение работы.**

Код программы представлен в приложении А.

1. При запуске программы пользователю выводится подсказка команды, которая выводит меню доступных команд и ожидается ввод команды с клавиатуры (рис. 1).



Рисунок . Запуск программы

1. Следующий шаг зависит от введенной команды, если пользователь ввёл:
   1. “0”, то выполнение программы завершается.
   2. “1”, то выводятся объёмы памяти, отводимые под различные типы данных со спецификатором (рис. 2)

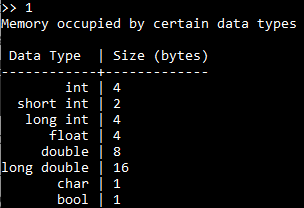


Рисунок . Задача 1

* 1. “2”, то ожидается ввод целого числа (int), двоичное представление которого потом будет выведено на экран (рис. 3).

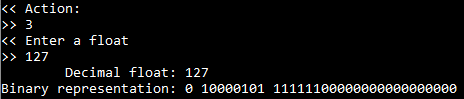


Рисунок . Задача 2, представление int в памяти

* 1. “3”, то ожидается ввод вещественного числа типа float, двоичное представление которого потом будет выведено на экран (рис. 4).

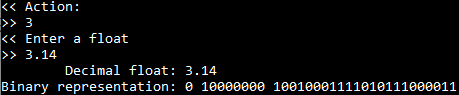
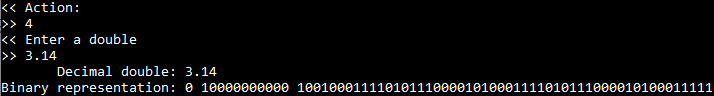
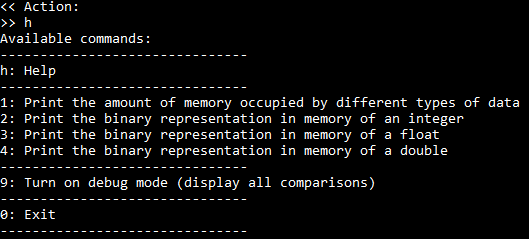


Рисунок . Задача 3, представление float в памяти

* 1. “4”, то ожидается ввод вещественного числа типа double, двоичное представление которого потом будет выведено на экран (рис. 5).

Рисунок . Задача 4, представление double в памяти

* 1. “h”, то выводится меню доступных команд (рис. 6).

Рисунок . Меню

* 1. Любой символ или набор символов, отличный от существующий команд, то будет выведено сообщение об ошибке (рис. 7).



Рисунок . Неизвестная команда

**Выводы.**

В ходе работы было изучено представление числовых типов данных в памяти компьютера, побитовые операции сдвига и поразрядной конъюнкции, методы перевода целых и вещественных чисел в двоичный код.

Приложение А

рабочий код

#include **<iostream>**#include **<iomanip>  
void** printSizeOfTypes() {  
 std::cout << **"\n Data Type | Size (bytes)\n"**;  
 std::cout << **"------------+-------------\n"**;  
 std::cout << **" int | "** << **sizeof**(**int**) << **'\n'**;  
 std::cout << **" short int | "** << **sizeof**(**short**) << **'\n'**;  
 std::cout << **" long int | "** << **sizeof**(**long**) << **'\n'**;  
 std::cout << **" float | "** << **sizeof**(**float**) << **'\n'**;  
 std::cout << **" double | "** << **sizeof**(**double**) << **'\n'**;  
 std::cout << **"long double | "** << **sizeof**(**long double**) << **'\n'**;  
 std::cout << **" char | "** << **sizeof**(**char**) << **'\n'**;  
 std::cout << **" bool | "** << **sizeof**(**bool**) << **'\n'**;  
}  
std::string getBitsOfInt(**int** value) {  
 */\*  
 \* Returns binary representation of int  
 \* :param value: int - decimal integer  
 \* :param isDebug: bool - displays every comparison if true  
 \* :return: std::string - binary representation of value  
 \*/* **unsigned int** order = **sizeof**(**int**) \* 8;  
 **unsigned int** mask = 1 << (order - 1);  
 std::string binaryInt;  
 **for** (**int** i = 1; i <= order; i++) {  
 **char** bit = value & mask ? **'1'** : **'0'**;  
 binaryInt += bit;  
 mask >>= 1;  
 **if** (i % 8 == 0 || i == 1)  
 binaryInt += **' '**;  
 }  
 **return** binaryInt;  
}  
std::string getBitsOfFloat(**float** input) {  
 */\*  
 \* Returns binary representation of float  
 \* :param input: float - decimal float  
 \* :param isDebug: bool - displays every comparison if true  
 \* :return: std::string - binary representation of input  
 \*/  
 // Use union due to binary operands isn't working with float* **union** FloatIntUnion {  
 **int** i;  
 **float** f;  
 };  
 FloatIntUnion value{.f = input};  
 **unsigned int** order = **sizeof**(**float**) \* 8;  
 **unsigned int** mask = 1 << (order - 1);  
 std::string binaryFloat;  
 **for** (**int** i = 1; i <= order; i++) {  
 **char** bit = value.i & mask ? **'1'** : **'0'**;  
 binaryFloat += bit;  
 mask >>= 1;  
 **if** (i == 1 || i == 9)  
 binaryFloat += **' '**;  
 }  
 **return** binaryFloat;  
}  
std::string getBitsOFDouble(**double** input) {  
 */\*  
 \* Returns binary representation of double  
 \* :param input: double - decimal double  
 \* :return: std::string - binary representation of input  
 \*/  
 // Use union due to binary operands isn't working with double  
 // Instead of integer which used in getBitsOfFloat use array of char  
 // One element of array represents one byte of double* **union** DoubleIntUnion {  
 **unsigned char** c[**sizeof**(**double**)];  
 **double** d;  
 };  
 DoubleIntUnion value{.d = input};  
 std::string binaryDouble;  
 **for** (**int** i = **sizeof**(**double**)-1; i >= 0; i--) {  
 **unsigned int** order = **sizeof**(**char**) \* 8;  
 **unsigned int** mask = 1 << (order - 1);  
 **for** (**int** j = 1; j <= order; j++) {  
 **char** bit = value.c[i] & mask ? **'1'** : **'0'**;  
 binaryDouble += bit;  
 mask >>= 1;  
 **int** currentBitIndex = (7 - i) \* 8 + j;  
 **if** (currentBitIndex == 1 || currentBitIndex == 12)  
 binaryDouble += **' '**;  
 }  
 }  
 **return** binaryDouble;  
}  
**int** main() {  
 std::cout.setf(std::ios::boolalpha);  
 std::cout << **"Enter 'h' to get list of commands\n"**;  
 *// Main loop* **while** (**true**) {  
 *// Get command number from user* **char** userAction;  
 std::cout << **"<< Action: \n>> "**;  
 std::cin.sync();  
 std::cin >> userAction;  
 *// Exit* **if** (userAction == **'0'**) {  
 **break**;  
 }  
 **switch** (userAction) {  
 *// Print the amount of memory occupied by different types of data* **case '1'**: {  
 std::cout << **"Memory occupied by certain data types\n"**;  
 printSizeOfTypes();  
 **break**;  
 }  
 *// Print the binary representation in memory of an integer* **case '2'**: {  
 **int** userInput;  
 std::cout << **"<< Enter an integer \n>> "**;  
 std::cin.sync();  
 std::cin >> userInput;  
 **if** (std::cin.fail()) {  
 std::cout << **"TypeError: invalid literal for int with base 10.\n"**;  
 std::cin.clear();  
 **break**;  
 }  
 std::cout << std::setw(23) << **"Decimal integer: "** << userInput << **'\n'**;  
 std::string binaryInt = getBitsOfInt(userInput);  
 std::cout << std::setw(23) << **"Binary representation: "** << binaryInt << **'\n'**;  
 **break**;  
 }  
 *// Print the binary representation in memory of a float* **case '3'**: {  
 **float** userInput;  
 std::cout << **"<< Enter a float \n>> "**;  
 std::cin.sync();  
 std::cin >> userInput;  
 **if** (std::cin.fail()) {  
 std::cout << **"TypeError: invalid literal for float with base 10.\n"**;  
 std::cin.clear();  
 **break**;  
 }  
 std::cout << std::setw(23) << **"Decimal float: "** << userInput << **'\n'**;  
 std::string binaryFloat = getBitsOfFloat(userInput);  
 std::cout << std::setw(23) << **"Binary representation: "** << binaryFloat << **'\n'**;  
 **break**;  
 }  
 *// Print the binary representation in memory of a double* **case '4'**: {  
 **double** userInput;  
 std::cout << **"<< Enter a double \n>> "**;  
 std::cin.sync();  
 std::cin >> userInput;  
 **if** (std::cin.fail()) {  
 std::cout << **"TypeError: invalid literal for double with base 10.\n"**;  
 std::cin.clear();  
 **break**;  
 }  
 std::cout << std::setw(23) << **"Decimal double: "** << userInput << **'\n'**;  
 std::string binaryDouble = getBitsOFDouble(userInput);  
 std::cout << std::setw(23) << **"Binary representation: "** << binaryDouble << **'\n'**;  
 **break**;  
 }  
 *// Help* **case 'h'**: {  
 std::cout << **"Available commands:\n"**;  
 std::cout << std::setw(32) << std::setfill(**'-'**) << **'\n'**;  
 std::cout << **"h: Help\n"**;  
 std::cout << std::setw(32) << std::setfill(**'-'**) << **'\n'**;  
 std::cout << **"1: Print the amount of memory occupied by different types of data\n"**;  
 std::cout << **"2: Print the binary representation in memory of an integer\n"**;  
 std::cout << **"3: Print the binary representation in memory of a float\n"**;  
 std::cout << **"4: Print the binary representation in memory of a double\n"**;  
 std::cout << std::setw(32) << std::setfill(**'-'**) << **'\n'**;  
 std::cout << **"0: Exit\n"**;  
 std::cout << std::setw(32) << std::setfill(**'-'**) << **'\n'**;  
 std::cout << std::setfill(**' '**);  
 **break**;  
 }  
 *// Unknown command* **default**: {  
 std::cout << **"RuntimeError: unknown command\n"**;  
 }  
 }  
 }  
 **return** 0;  
}