**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

отчет

**по лабораторной работе № 1**

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

Тема: Исследование внутреннего представления различных форматов данных

|  |  |  |
| --- | --- | --- |
| Студент гр. 3311 |  | Баймухамедов Р. Р. |
| Преподаватель |  | Гречухин М. Н. |

Санкт-Петербург

2024

**Цель работы.**

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

**Задание (вариант 3)**

1. Разработать алгоритм ввода с клавиатуры unsinged char и float число и показать на экране его внутреннее представление в двоичной системе счисления
2. Написать и отладить программу на языке C++, реализующую разработанный алгоритм. Программа должна

* Иметь дружественный интерфейс
* Выводить на экран информативное сообщение при вводе некорректных данных
* Предложить повторный ввод пока не будут введены корректные данные

1. Установить в заданное пользователем состояние определённое количество рядом стоящих бит, номер старшего бита, как и всё остальное, вводится с клавиатуры.

**Постановка задачи и описание решения**

Для вывода двоичного представления unsigned char используются побитовые операции. Задаётся переменная, имеющая значение 1, а затем происходит побитовое сравнение с unsigned char и при его равенстве выводится единица, иначе ноль. Далее при помощи битовых операции производим сдвиг переменной, имеющей значение 1, влево, и так до тех пор, пока не будут проверены все биты числа.

Для вывода двоичного представления числа типа float также используются побитовые операции. Однако, в отличие от простого целого типа, для работы с битами числа с плавающей запятой требуется доступ к его внутреннему представлению, так как оно хранится в формате IEEE 754 (1 бит знака, 8 бит экспоненты, 23 бита мантиссы).

Для этого используется union, который позволяет хранить одно и то же значение в разных типах данных. Мы записываем значение типа float в переменную union и можем получить доступ к его битам через эквивалентное представление типа uint32\_t. Далее происходит побитовое сравнение с каждой позицией бита, начиная с самого старшего, с помощью сдвига битов влево. Если бит установлен, выводится единица, если нет — ноль. Пробелы вставляются для разделения различных частей числа (знак, экспонента и мантисса).

Для установки определённого количества битов в заданное пользователем состояние используется побитовая маска.

Сначала пользователь задаёт номер старшего бита и количество рядом стоящих битов, которые нужно изменить. На основе этих данных строится маска. Сначала сдвигаем единицу влево на количество битов, чтобы получить число с единицами на нужных позициях. Затем это число сдвигается влево ещё раз на столько бит, чтобы единицы оказались в нужной части числа.

Если необходимо установить биты в 1, то используется операция побитового ИЛИ (|), которая устанавливает биты в единицу на позициях, где в маске стоит 1. Если нужно установить биты в 0, применяется побитовая инверсия маски с помощью операции NOT (~), а затем используется побитовое И (&), чтобы сбросить нужные биты.

Пример маски:

Входные данные: старший бит — 4, кол-во битов — 2, состояние — 1

Маска будет: 00011100 для изменения двух битов, начиная с 4-го.

Хранение данных вещественного типа описывается IEEE 754 - стандартом в виде значения мантиссы (M) со знаком (S) и значения порядка (P). Число бит для хранения мантиссы и порядка зависит от типа данных с плавающей запятой.

float

|  |  |  |
| --- | --- | --- |
| S | P | M |

31 30 23 22 0

double

|  |  |  |
| --- | --- | --- |
| S | P | M |

63 62 52 51 0

long double

|  |  |  |
| --- | --- | --- |
| S | P | M |

79 78 64 63 0

Вещественное число в памяти хранится с нормализованной мантиссой, значение которой в десятичном эквиваленте лежит в диапазоне от 1 до 2. Причём 2 не входит в границу диапазона. Если в процессе выполнения какой-либо операции над данными с плавающей запятой значение мантиссы выходит из указанного диапазона, то в конце операции выполняется нормализация результата путем приведения значения мантиссы к указанному диапазону с соответствующим изменением значения порядка.

Порядок числа в соответствии с указанным форматом хранится «сдвинутым», то есть к его действительному значению добавляется в зависимости от формата такое число, чтобы порядок Р был всегда неотрицательным. Для формата float прибавляется 127. Всегда неотрицательный порядок упрощает выполнение операции сравнения порядков и арифметических операций над ними, а также избавляет от необходимости выделять один бит для хранения знака порядка

**Контрольные примеры**

**Пример 1:**

Исходные данные:

uchar\_value = 64

hob = 4

count = 2

state = 1

Результаты:

(Original data)

Value as symbol: @

Value as number: 64

Value in binary system: 01000000

(…) Task Algorithm

Value as symbol: \

Value as number: 92

Value in binary system: 01011100

**Пример 2:**

Исходные данные:

float\_value = 13.25

hob = 21

count = 4

state = 1

Результаты:

(Original data)

Value as number: 13.25

Value in binary system: 0 10000010 10101000000000000000000

(…) Task Algorithm

Value as number: 15.875

Value in binary system: 0 10000010 11111100000000000000000

**Пример 3:**

Исходные данные:

uchar\_value = 127

hob = 4

count = 2

state = 0

Результаты:

(Original data)

Value as symbol:

Value as number: 127

Value in binary system: 01111111

(…) Task Algorithm

Value as symbol: c

Value as number: 99

Value in binary system: 01100011

**Текст программы**

#include <iostream>

#include <climits>

#include <limits>

#include <string>

#include <cstdint>

// Union

union float\_union{

float float\_value;

uint32\_t bit\_value;

};

// Input

int enter\_integer(const std::string& message, int a, int b) {

int input;

bool flag = false;

do {

std::cout << message;

std::cin >> input; // cause input is int then will be read only first integer part of input

if (input >= a && input <= b && std::cin.peek() == '\n') { // peek() to check has something else after integer part

flag = true;

} else {

std::cout << "Entered value is not correct. Please try again.Entered value should be in [" << a << ", " << b << "]" << std::endl;

std::cin.clear(); // reset all flags of errors to accept input again

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); // ignore and clear all other symbols

}

} while (!flag);

return input;

}

float enter\_float(const std::string& message, float a, float b) {

float input;

bool flag = false;

do {

std::cout << message;

std::cin >> input; // cause input is float then will be read only first float part of input

if (input >= a && input <= b && std::cin.peek() == '\n') { // peek() to check has something else after float part

flag = true;

} else {

std::cout << "Entered value is not correct. Please try again. Entered value should be in [" << a << ", " << b << "]" << std::endl;

std::cin.clear(); // reset all flags of errors to accept input again

std::cin.ignore(std::numeric\_limits<std::streamsize>::max(), '\n'); // ignore and clear all other symbols

}

} while (!flag);

return input;

}

unsigned char input\_unsigned\_char(){

int input = enter\_integer("Enter the value (unsigned char): ", 0, UCHAR\_MAX);

return static\_cast<unsigned char>(input);

}

float input\_float(){

float input = enter\_float("Etner the value (float): ", std::numeric\_limits<float>::lowest(), std::numeric\_limits<float>::max());

return input;

}

// Output

void print\_unsigned\_char(unsigned char value){

std::cout << "\nValue as symbol: " << value << "\n";

std::cout << "Value as number: " << static\_cast<int>(value) << "\n";

std::cout << "Value in binary system: ";

for(int i=sizeof(char)\*8-1;i>=0;i--){ // sizeof return value in bytes

if(value & (1<<i)) std::cout << 1;

else std::cout << 0;

}

std::cout << std::endl;

}

void print\_float(float value){

std::cout << "\nValue as number: " << value << "\n";

float\_union bits;

bits.float\_value = value; // fill `union bits` bites of float value

std::cout << "Value in binart system: ";

for(int i=sizeof(float)\*8-1; i>=0;i--){ // sizeof return value in bytes

if(bits.bit\_value & (1<<i)) std::cout << 1;

else std::cout << 0;

if(i==31) std::cout << " "; // to divide sign bite

else if(i==23) std::cout << " "; // to divide `p` and `m`

}

std::cout << std::endl;

}

// Task

void unsigned\_char\_task(unsigned char &value){

std::cout << "\n";

int hob = enter\_integer("Enter the number of high-order bit: ", 0, sizeof(char)\*8-1); // for example 4

int count = enter\_integer("Enter the count of changable bits: ", 0, hob); // for example 2

int state = enter\_integer("Enter the state of bits: ", 0, 1); // for example 1

unsigned char mask = ((1<<count+1)-1)<<(hob-count); //step by step: 00000001 -> 00001000 -> 00000111 -> 00011100

if(state) value |= mask;

else value &= ~mask; // if state equial zero, then mask transfrom to 11100011

}

void float\_task(float &value){

std::cout << "\n";

int hob = enter\_integer("Enter the number of high-order bit: ", 0, sizeof(float)\*8-1); // for exmaple 4

int count = enter\_integer("Enter the count of changable bits: ", 0, hob); // for example 2

int state = enter\_integer("Enter the state of bits: ", 0, 1); // for example 1

float\_union bits;

bits.float\_value = value; // fill `union bits` bites of float value

int mask = ((1<<count+1)-1)<<(hob-count); //step by step: 000...00001 -> 000...01000 -> 000...00111 -> 000...11100

if(state) bits.bit\_value |= mask;

else bits.bit\_value &= ~mask; // invert mask

value = bits.float\_value; // bites of `value` take bites of `union bits`

}

int main(){

int option=0;

do{

std::cout << "Choose the option:\n0 - for exit\n1 - for unsigned char\n2 - for float\n";

option = enter\_integer("Enter the option: ",0, 2);

switch(option){

case 1: {

unsigned char uchar\_value = input\_unsigned\_char();

print\_unsigned\_char(uchar\_value);

unsigned\_char\_task(uchar\_value);

print\_unsigned\_char(uchar\_value);

std::cout << "\n\_\_\_\_\_\_\_\_\_\_\n\n";

break;

}

case 2: {

float float\_value = input\_float();

print\_float(float\_value);

float\_task(float\_value);

print\_float(float\_value);

std::cout << "\n\_\_\_\_\_\_\_\_\_\_\n\n";

break;

}

case 0:

option = 0;

break;

default:

break;

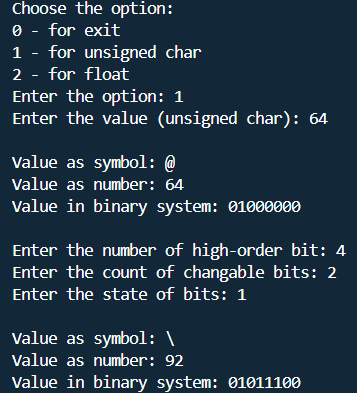
}

} while(option!=0);

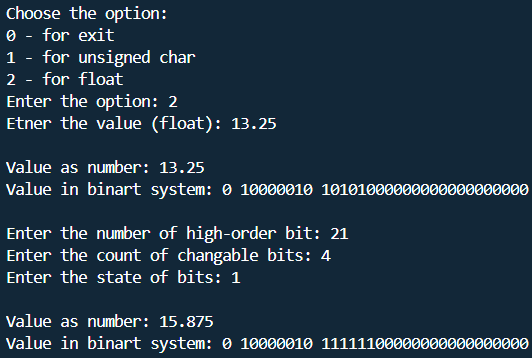
return 0;

}

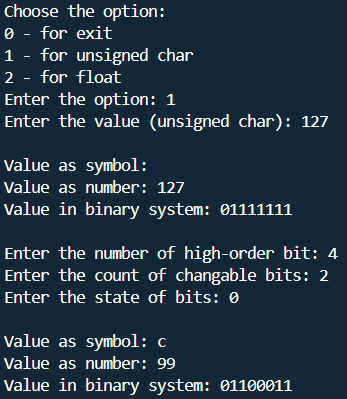
**Примеры выполнения программы**

****

Пример 1

****

Пример 2

****

Пример 3

**Выводы.**

В ходе выполнения работы мы изучили, как компьютер хранит и обрабатывает различные типы данных, а также подробно рассмотрели двоичное представление целых чисел и операции сдвига битов. Мы узнали и применили следующие ключевые моменты:

Мы изучили, как целые числа хранятся в памяти компьютера в виде двоичных кодов. Для этого было реализовано отображение чисел в двоичном виде, что позволило визуализировать, как каждый бит числа используется для представления данных. Также мы научились работать с побитовыми сдвигами чисел.

В процессе работы мы коснулись темы представления чисел с плавающей запятой, где изучили структуру числа: знак, экспонента и мантисса. Это дало понимание, как числа с плавающей запятой хранятся в двоичной системе.

Был разработан и реализован алгоритм, который позволяет пользователю старший бит, а также количество рядом стоящих битов и состояние, на которое эти биты поменяются.

Таким образом, мы научились эффективно работать с двоичными представлениями данных, узнали, как компьютер хранит целые числа и числа с плавающей запятой, и реализовали операции сдвига и перестановки битовых групп с учётом внутренней структуры данных.