**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 гр. 2372 |  | Соколовский В.Д. |
| Преподаватель |  | Гречухин М.Н. |

Санкт-Петербург

2023

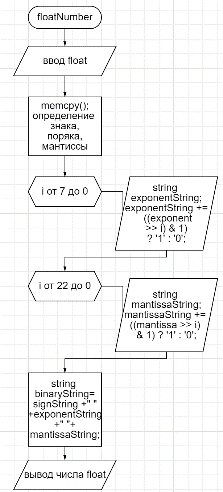
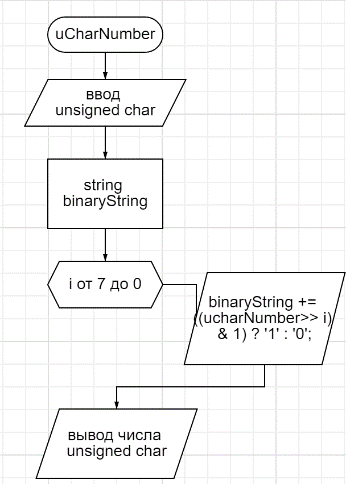
**Цель работы.**

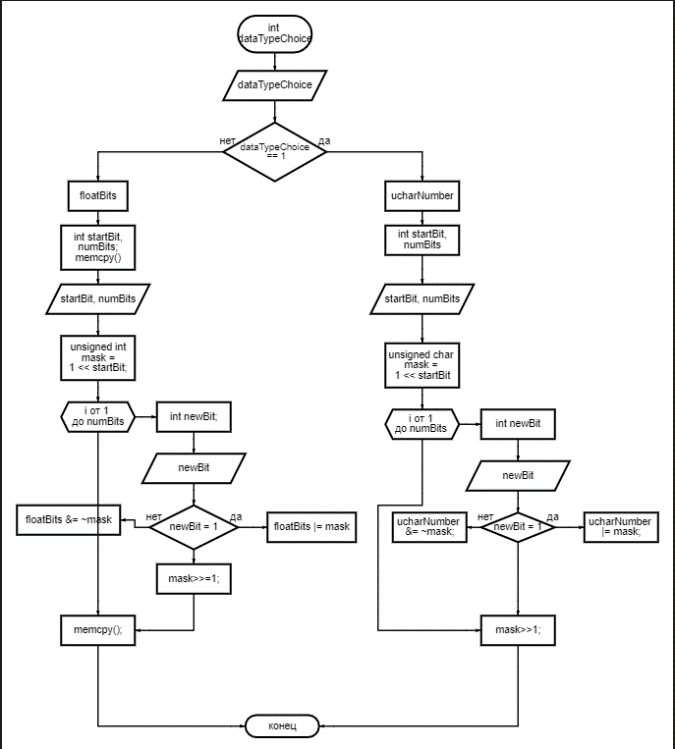
Знакомство с внутренним представлением различных типов данных, используемых компьютером при их обработке.

**Задание на лабораторную работу.**

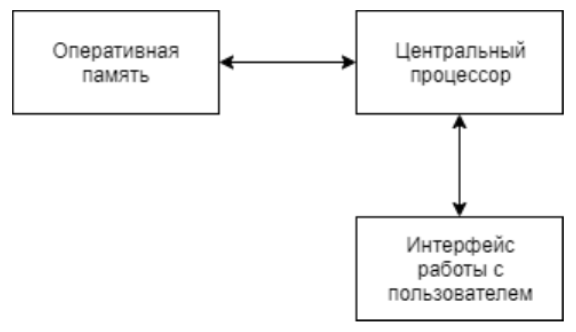
1. Разработать алгоритм ввода с клавиатуры типов данных(unsigned char, float) и показать на экране их внутреннее представление в двоичной системе счисления.
2. Установить в заданное пользователем состояние определённое количество рядом стоящих бит, номер старшего бита, как и всё остальное, вводится с клавиатуры.

**Блок-схема алгоритма**

** **

****

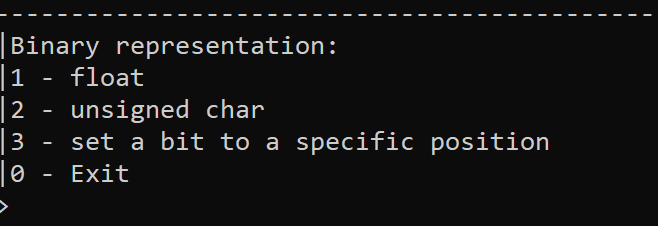
**Структурная схема аппаратных средств**



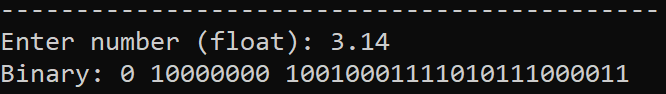
**Пример запуска программы**

1. Следующий шаг зависит от введенной команды, если пользователь ввёл:

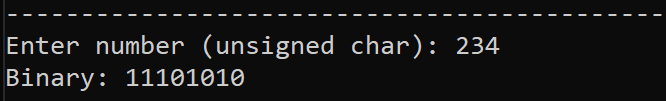
Выбор (вводить с клавиатуры):



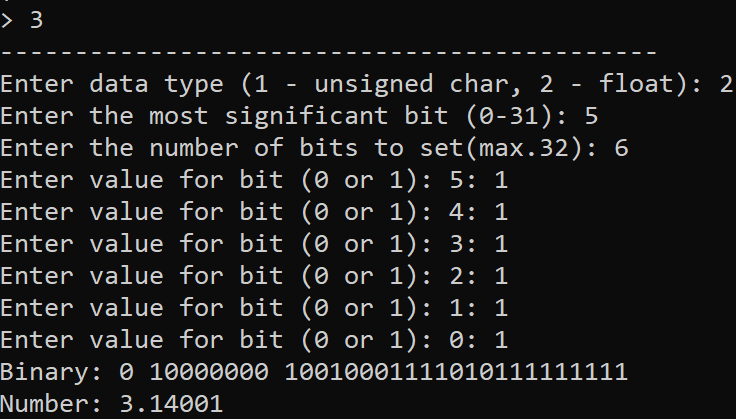
1. «1» - Ввод числа типа float и вывод его в двоичной форме



1. «2» - Ввод числа типа unsigned char и вывод его в двоичной форме



1. «3» - Поменять местами рядом стоящие биты



**Выводы.**

**В ходе данной лабораторной работы ознакомились с внутренним представлением различных типов данных, используемых компьютером при их обработке.**

Приложение А

рабочий код

**application.h**

#ifndef PRACTICAL1\_APPLICATION\_H  
#define PRACTICAL1\_APPLICATION\_H  
  
#include "internalrepres.h"  
  
class application {  
public:  
 int exec();  
private:  
 int menu();  
};  
  
  
#endif //PRACTICAL1\_APPLICATION\_H

application.cpp

#include "application.h"  
#include "internalrepres.h"  
  
#include <iostream>  
#include <limits>  
  
using namespace std;  
  
  
int application::exec() {  
  
 unsigned char ucharNumber = 0;  
 float floatNumber = 0.0f;  
  
 char ch;  
 bool exit = false;  
 while(!exit){  
 ch = menu();  
 switch(ch){  
 case '1':{  
 cout << "Enter number (float): ";  
 if(!(cin >> floatNumber)){  
 cin.clear();  
 cin.ignore(numeric\_limits<streamsize>::max(), '\n');  
 cerr << "Invalid" << endl;  
 break;  
 }  
  
 string binaryRepresentation = internalrepres::FloatToBinary(floatNumber);  
 cout << "Binary: " << "" << binaryRepresentation << endl;  
 break;  
 }  
 case '2':{  
 unsigned int inputUcharNumber;  
 cout << "Enter number (unsigned char): ";  
 if(!(cin >> inputUcharNumber)){  
 cin.clear();  
 cin.ignore(numeric\_limits<streamsize>::max(), '\n');  
 cerr << "Invalid" << endl;  
 break;  
 }  
 else{  
 ucharNumber = static\_cast<unsigned char>(inputUcharNumber);  
 string binaryRepresentation = internalrepres::UnsignedCharToBinary(ucharNumber);  
 cout << "Binary: "<< "" << binaryRepresentation << endl;  
 }  
 break;  
 }  
 case '3':{  
 int dataTypeChoice;  
 cout << "Enter data type (1 - unsigned char, 2 - float): ";  
 cin >> dataTypeChoice;  
 if(dataTypeChoice == 1){  
 internalrepres::SetBitsU(ucharNumber);  
 string binaryRepresentation = internalrepres::UnsignedCharToBinary(ucharNumber);  
 cout << "Binary: "<< "" << binaryRepresentation << endl;  
 cout << "Number: " << "" << static\_cast<int>(ucharNumber) << endl;  
 }  
 else if(dataTypeChoice == 2){  
 internalrepres::SetBitsF(floatNumber);  
 string binaryRep = internalrepres::FloatToBinary(floatNumber);  
 cout << "Binary: " << "" << binaryRep << endl;  
 cout << "Number: " << "" << static\_cast<float>(floatNumber) << endl;  
 }  
 else{  
 cout << "Incorrect data type" << endl;  
 }  
 break;  
 }  
 case '0':{  
 std::cout << "Exit" << std::endl;  
 exit = true;  
 break;  
 }  
 default:{  
 cout << "Error\n";  
 break;  
 }  
 }  
 }  
}  
  
  
  
int application::menu(){  
 char ch;  
 cout << "--------------------------------------------" << endl;  
 cout << "|Binary representation:" << endl;  
 cout << "|1 - float" << endl;  
 cout << "|2 - unsigned char" << endl;  
 cout << "|3 - set a bit to a specific position" << endl;  
 cout << "|0 - Exit" << endl << "> ";  
 cin >> ch;  
 cout << "--------------------------------------------" << endl;  
 return ch;  
}

main.cpp

#include "application.h"  
  
int main(){  
  
 application a;  
  
 a.exec();  
 return 0;  
}

internalrepres.cpp

#include "internalrepres.h"  
#include <iostream>  
  
  
  
string internalrepres::FloatToBinary(float floatNumber) {  
  
 unsigned int floatBits;  
 memcpy(&floatBits, &floatNumber, sizeof(floatNumber));  
 /\* memcpy - for bitwise representation of numbers  
 \* &floatBits - pointer to the memory area into which the binary representation floatNumber is copied  
 \* &floatNumber - pointer to the memory area from which the binary representation of floatNumber was copied  
 \* sizeof(floatNumber) - size in bytes \*/  
  
 int sign = (floatBits >> 31) & 1;  
 int exponent = (floatBits >> 23) & 0xFF; // 0xFF - 255 (IEEE754)  
 int mantissa = floatBits & 0x7FFFFF;  
  
 string signString = to\_string(sign);  
 string exponentString;  
 for(int i = 7; i >= 0; --i){  
 exponentString += ((exponent >> i) & 1) ? '1' : '0';  
 }  
 string mantissaString;  
 for(int i = 22; i >= 0; --i){  
 mantissaString += ((mantissa >> i) & 1) ? '1' : '0';  
 }  
  
 string binaryString = signString + " " + exponentString + " " + mantissaString;  
  
 return binaryString;  
}  
  
  
  
string internalrepres::UnsignedCharToBinary(unsigned char ucharNumber) {  
  
 string binaryString;  
  
 for(int i = 7; i >= 0; --i){  
 binaryString += ((ucharNumber >> i) & 1) ? '1' : '0';  
 }  
  
 return binaryString;  
}  
  
  
  
void internalrepres::SetBitsU(unsigned char& ucharNumber) {  
  
 int startBit, numBits;  
  
 cout << "Enter the most significant bit (0-7): ";  
 cin >> startBit;  
  
 cout << "Enter the number of bits to set(max.8): ";  
 cin >> numBits;  
  
 if(startBit < 0|| startBit >= 8 || numBits <= 0 || numBits > 8 || startBit - numBits + 1 < 0){  
 cout << "Invalid" << endl;  
 return;  
 }  
  
 unsigned char mask = 1 << startBit;  
  
 for(int i = 0; i < numBits; ++i){  
 cout << "Enter value for bit (0 or 1): " << (startBit - i) << ": ";  
 int newBit;  
 cin >> newBit;  
 if(newBit != 0 && newBit != 1){  
 cout << "Enter a valid value (0 or 1) for the bit" << endl;  
 return;  
 }  
 if(newBit == 1){  
 ucharNumber |= mask;  
 }  
 else{  
 ucharNumber &= ~mask; //inversion, changes from 0 to 1 and from 1 to 0  
 }  
 mask >>= 1;  
 }  
}  
  
  
  
void internalrepres::SetBitsF(float& floatNumber) {  
  
 unsigned int floatBits;  
 memcpy(&floatBits, &floatNumber, sizeof(floatNumber));  
  
 int startBit, numBits;  
  
 cout << "Enter the most significant bit (0-31): ";  
 cin >> startBit;  
  
 cout << "Enter the number of bits to set(max.32): ";  
 cin >> numBits;  
  
 if(startBit < 0 || startBit >= 32 || numBits <= 0 || numBits > 32 || startBit - numBits + 1 < 0){  
 cout << "Invalid" << endl;  
 return;  
 }  
  
 unsigned int mask = 1 << startBit;  
  
 for(int i = 0; i < numBits; ++i){  
 cout << "Enter value for bit (0 or 1): " << (startBit - i) << ": ";  
 int newBit;  
 cin >> newBit;  
 if(newBit != 0 && newBit != 1){  
 cout << "Enter a valid value (0 or 1) for the bit" << endl;  
 return;  
 }  
 if(newBit == 1){  
 floatBits |= mask;  
 }  
 else{  
 floatBits &= ~mask;  
 }  
 mask >>= 1;  
 }  
  
 memcpy(&floatNumber, &floatBits, sizeof(floatNumber));  
}

interalrepres.h

#ifndef PRACTICAL1\_INTERNALREPRES\_H  
#define PRACTICAL1\_INTERNALREPRES\_H  
  
#include <string>  
  
using namespace std;  
  
class internalrepres {  
public:  
 static string FloatToBinary(float floatNumber);  
 static string UnsignedCharToBinary(unsigned char ucharNumber);  
 static void SetBitsU(unsigned char& ucharNumber);  
 static void SetBitsF(float& floatNumber);  
};  
  
#endif //PRACTICAL1\_INTERNALREPRES\_H