

**МИНОБРНАУКИ РОССИИ  
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ  
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)  
КАФЕДРА ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ**

**ОТЧЁТ**  
**по лабораторной работе №1**  
**по дисциплине «Организация ЭВМ и систем»**  
**Тема: ИССЛЕДОВАНИЕ ВНУТРЕННЕГО ПРЕДСТАВЛЕНИЯ**  
**РАЗЛИЧНЫХ ФОРМАТОВ ДАННЫХ**

Студент гр. 3312	_____	Барченков П.А.
Студент гр. 3312	_____	Лебедев И.А.
Студент гр. 3312	_____	Шарапов И.Д.
Преподаватель	_____	Ильин С.Е.

Санкт-Петербург  
2024

## Содержание

Описание задания .....	3
Схема алгоритма.....	4
Текст программы .....	8
Примеры запуска программы .....	10
Приложение .....	12

## Описание задания

Вариант 9.

1. Разработать алгоритм ввода с клавиатуры типов данных (*short int* и *float*) и показать на экране их 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.

3. Дополнить разработанный ранее алгоритм возможностью инвертировать значения всех бит кроме тех, количество и номера которых задаются с клавиатуры и последующего вывода преобразованного кода в двоичной системе счисления и в формате исходного данного.

## Схема алгоритма

Блок кода, представленный на рисунке 1, демонстрирует работу функции main, в которой мы считываем, обрабатываем и выводим числа. Сначала проводим полную обработку ShortInt, затем Float.

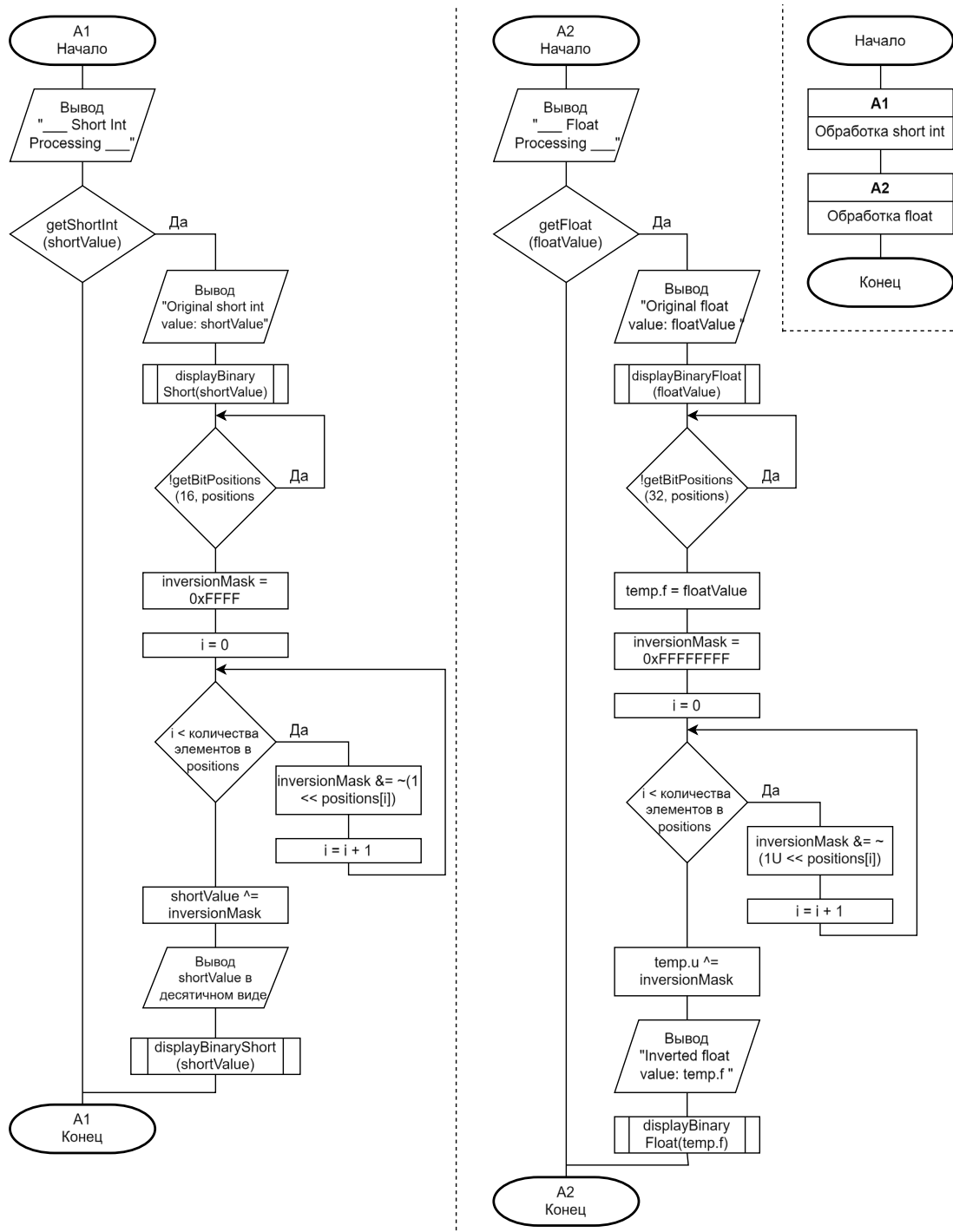


Рисунок 1 – Схема функции main

Блок кода, представленный на рисунке 2, демонстрирует работу функции getBitPosition, в которой происходит обработка некорректного ввода

пользователем и считывание индексов тех битов, которые не нужно инвертировать.

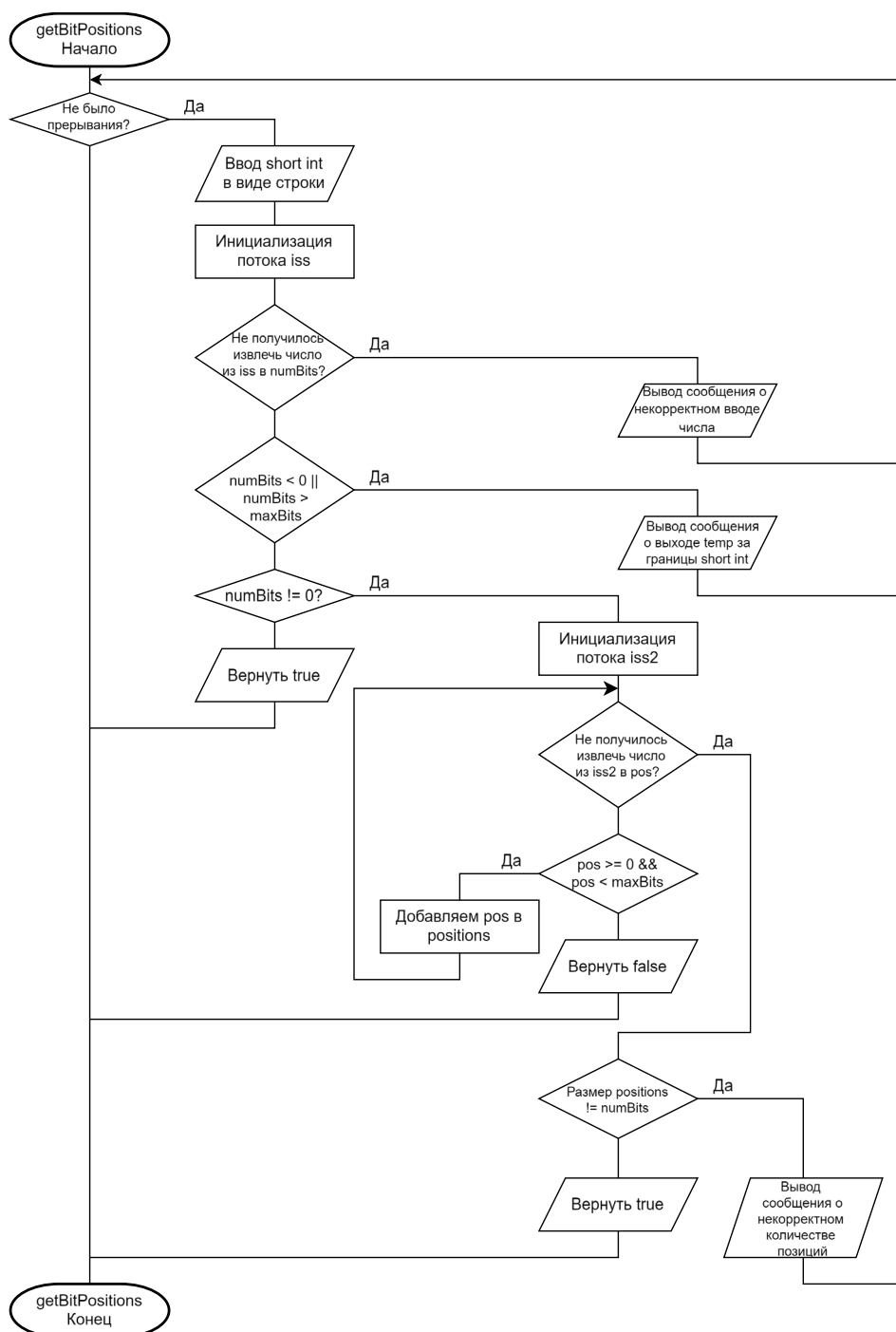


Рисунок 2 – Схема функции getBitPosition

Блок кода, представленный на рисунке 3, демонстрирует работу двух функций для обработки ShortInt. getShortInt используется для обработки некорректного ввода пользователем и считыванием самого числа. displayBinaryShort для вывода на экран двоичного представления полученного числа.

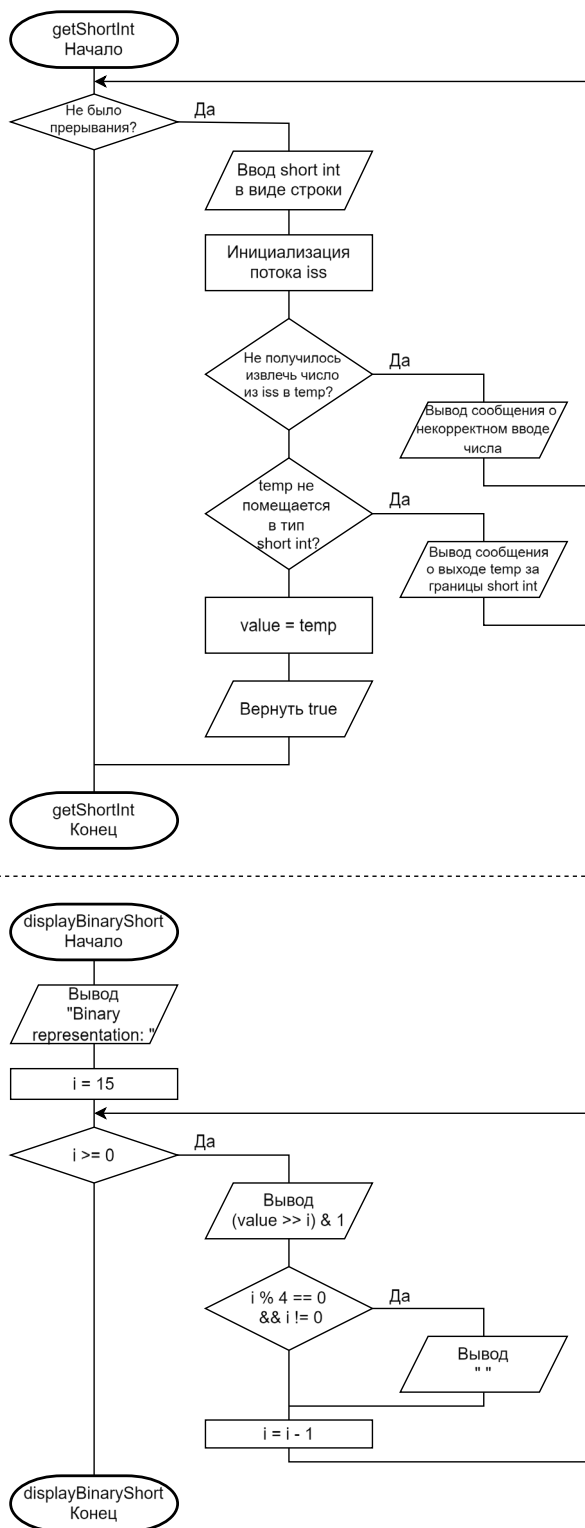


Рисунок 3 – Схема функций для работы с ShortInt

Блок кода, представленный на рисунке 4, демонстрирует работу двух функций для обработки Float. getFloat используется для обработки некорректного ввода пользователем и считыванием самого числа. displayBinaryFloat для вывода на экран двоичного представления полученного числа.

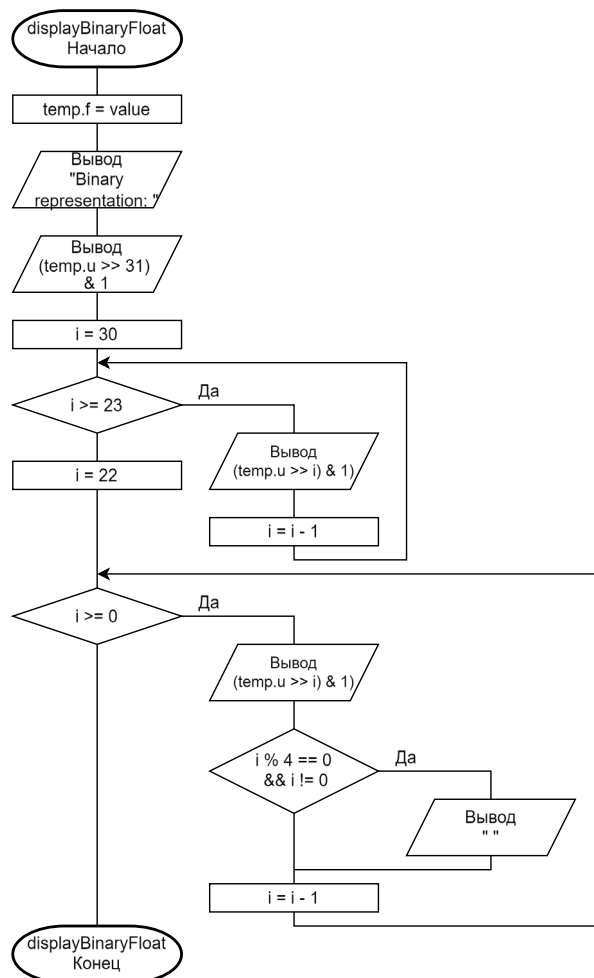
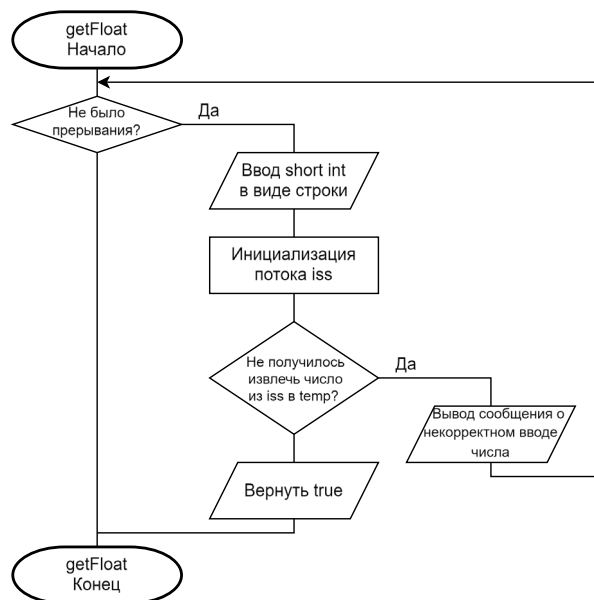


Рисунок 4 - Схема функций для работы с Float

## Текст программы

```
#include <iostream>
#include <limits>
#include <string>
#include <sstream>
#include <vector>

void displayBinaryShort(short int value) {
    std::cout << "Binary representation: ";
    for (int i = 15; i >= 0; --i) {
        std::cout << ((value >> i) & 1);
        if (i % 4 == 0 && i != 0) std::cout << " ";
    }
    std::cout << std::endl;
}

void displayBinaryFloat(float value) {
    union {
        float f;
        unsigned int u;
    } temp;
    temp.f = value;

    std::cout << "Binary representation: ";

    // Print the sign bit (1 bit)
    std::cout << ((temp.u >> 31) & 1) << " ";

    // Print the exponent bits (8 bits)
    std::cout << "(";
    for (int i = 30; i >= 23; --i) {
        std::cout << ((temp.u >> i) & 1);
    }
    std::cout << ") ";

    // Print the mantissa bits (23 bits)
    std::cout << "(";
    for (int i = 22; i >= 0; --i) {
        std::cout << ((temp.u >> i) & 1);
        if (i % 4 == 0 && i != 0) {
            std::cout << " ";
        }
    }
    std::cout << ")" << std::endl;
}

bool getShortInt(short int & value) {
    while (true) {
        std::cout << "Please enter a short int value (from -32768 to 32767):";
        std::string input;
        std::getline(std::cin, input);

        std::istringstream iss(input);
        int temp;
        if (iss >> temp) {
            if (temp >= std::numeric_limits<short int>::min() &&
                temp <= std::numeric_limits<short int>::max()) {
                value = static_cast<short int>(temp);
                return true;
            } else {
                std::cout << "Value out of range for short int. Please try again (from -32768 to 32767)." << std::endl;
            }
        } else {
            std::cout << "Invalid input. Please enter a valid short int number." << std::endl;
        }
    }
}

bool getFloat(float & value) {
    while (true) {
        std::cout << "Please enter a float value:";
        std::string input;
        std::getline(std::cin, input);

        std::istringstream iss(input);
        if (iss >> value) {
```



```

        return true;
    } else {
        std::cout << "Invalid input. Please enter a valid float number." << std::endl;
    }
}

bool getBitPositions(int maxBits, std::vector<int>& positions) {
    while (true) {
        positions.clear();
        std::cout << "Please enter the number of bits to exclude from inversion (0 to " << maxBits <<
        "):";
        std::string input;
        std::getline(std::cin, input);
        std::istringstream iss(input);
        int numBits;
        if (iss >> numBits) {
            if (numBits >= 0 && numBits <= maxBits) {
                if (numBits == 0) return true;
                std::cout << "Enter the bit positions to exclude (separated by spaces, between 0 and "
                << (maxBits - 1) << "):";
                std::getline(std::cin, input);
                std::istringstream iss2(input);
                int pos;
                while (iss2 >> pos) {
                    if (pos < 0 || pos >= maxBits) {
                        std::cout << "Bit position " << pos << " is out of range. Please enter
                        positions between 0 and " << (maxBits - 1) << "." << std::endl;
                        return false;
                    }
                    positions.push_back(pos);
                }
                if (positions.size() != static_cast<size_t>(numBits)) {
                    std::cout << "Number of positions entered does not match the number specified.
                    Please try again." << std::endl;
                    continue;
                }
                return true;
            } else {
                std::cout << "Invalid number of bits to exclude. Please enter a number between 0 and "
                << maxBits << "." << std::endl;
            }
        } else {
            std::cout << "Invalid input. Please enter a valid number." << std::endl;
        }
    }
}

int main() {
    std::cout << "    Short Int Processing    " << std::endl;
    short int shortValue;
    if (getShortInt(shortValue)) {
        std::cout << "Original short int value: " << shortValue << std::endl;
        displayBinaryShort(shortValue);

        std::vector<int> positions;
        while (!getBitPositions(16, positions)) {

        }

        unsigned short inversionMask = 0xFFFF;
        for (size_t i = 0; i < positions.size(); ++i) {
            inversionMask &= ~(1 << positions[i]);
        }

        shortValue ^= inversionMask;
        std::cout << "Inverted short int value: " << shortValue << std::endl;
        displayBinaryShort(shortValue);
    }

    std::cout << "\n    Float Processing    " << std::endl;
    float floatValue;
    if (getFloat(floatValue)) {
        std::cout << "Original float value: " << floatValue << std::endl;
        displayBinaryFloat(floatValue);

        std::vector<int> positions;
        while (!getBitPositions(32, positions)) {
    }

```

```

union {
    float f;
    unsigned int u;
} temp;
temp.f = floatValue;

unsigned int inversionMask = 0xFFFFFFFF;
for (size_t i = 0; i < positions.size(); ++i) {
    inversionMask &= ~(1U << positions[i]);
}

temp.u ^= inversionMask;
std::cout << "Inverted float value: " << temp.f << std::endl;
displayBinaryFloat(temp.f);
}

return 0;
}

```

## Примеры запуска программы

В первом примере (Рис. 5) демонстрируется обработка ошибочного ввода исходного числа пользователем.

```

___ Short Int Processing ___
Please enter a short int value (from -32768 to 32767):80000
Value out of range for short int. Please try again (from -32768 to 32767).
Please enter a short int value (from -32768 to 32767):239
Original short int value: 239
Binary representation: 0000 0000 1110 1111
Please enter the number of bits to exclude from inversion (0 to 16):3
Enter the bit positions to exclude (separated by spaces, between 0 and 15):11 0 7
Inverted short int value: -2159
Binary representation: 1111 0111 1001 0001

___ Float Processing ___
Please enter a float value:93.2
Original float value: 93.2
Binary representation: 0 (10000101) (011 1010 0110 0110 0110)
Please enter the number of bits to exclude from inversion (0 to 32):50
Invalid number of bits to exclude. Please enter a number between 0 and 32.
Please enter the number of bits to exclude from inversion (0 to 32):0
Inverted float value: -0.0482422
Binary representation: 1 (01111010) (100 0101 1001 1001 1001 1001)

Process finished with exit code 0

```

Рисунок 5 – Пример с ошибочным вводом в первом поле

Во втором примере (Рис. 6) демонстрируется обработка большого числа, неправильного ввода индексов пользователем и вводом букв там, где требуется число.

```

___ Short Int Processing ___
Please enter a short int value (from -32768 to 32767):17
Original short int value: 17
Binary representation: 0000 0000 0001 0001
Please enter the number of bits to exclude from inversion (0 to 16):5
Enter the bit positions to exclude (separated by spaces, between 0 and 15):2 3 9 13 15
Inverted short int value: 24034
Binary representation: 0101 1101 1110 0010

___ Float Processing ___
Please enter a float value:10000000000000000000000000000000
Original float value: 1e+27
Binary representation: 0 (11011000) (100 1110 1100 1011 1000 1111)
Please enter the number of bits to exclude from inversion (0 to 32):1
Enter the bit positions to exclude (separated by spaces, between 0 and 31):32
Bit position 32 is out of range. Please enter positions between 0 and 31.
Number of positions entered does not match the number specified. Please try again.
Please enter the number of bits to exclude from inversion (0 to 32):fsd
Invalid input. Please enter a valid number.
Please enter the number of bits to exclude from inversion (0 to 32):311
Enter the bit positions to exclude (separated by spaces, between 0 and 31):3
Number of positions entered does not match the number specified. Please try again.
Please enter the number of bits to exclude from inversion (0 to 32):1
Enter the bit positions to exclude (separated by spaces, between 0 and 31):31
Inverted float value: 4.47328e-27
Binary representation: 0 (00100111) (011 0001 0011 0100 0111 0000)

```

Рисунок 6 – Пример с ошибным вводом во втором и третьем поле

В третьем примере (Рис. 7) демонстрируется работа программы с полностью корректными данными, введенными пользователем.

```

___ Short Int Processing ___
Please enter a short int value (from -32768 to 32767):2129
Original short int value: 2129
Binary representation: 0000 1000 0101 0001
Please enter the number of bits to exclude from inversion (0 to 16):3
Enter the bit positions to exclude (separated by spaces, between 0 and 15):15 4 7
Inverted short int value: 30526
Binary representation: 0111 0111 0011 1110

___ Float Processing ___
Please enter a float value:21.29
Original float value: 21.29
Binary representation: 0 (10000011) (010 1010 0101 0001 1110 1100)
Please enter the number of bits to exclude from inversion (0 to 32):3
Enter the bit positions to exclude (separated by spaces, between 0 and 31):15 4 7
Inverted float value: -0.208185
Binary representation: 1 (01111100) (101 0101 0010 1110 1000 0011)

Process finished with exit code 0

```

Рисунок 7 – Пример без ошибок

## **Приложение**

В примерах программа выполнялась в среде разработки JetBrains CLion 2024.2.1., стандартом языка C++ 98 и под управлением ОС Linux Mint 22.