**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САПР**

**отчет**

**по лабораторной работе №1**

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

**Тема: «Исследование внутреннего представления различных форматов данных»**

**Вариант 5**

|  |  |  |
| --- | --- | --- |
| Студент гр. 0302 |  | Чёрный Я.И. |
| Преподаватель |  | Жандаров В.В. |

Санкт-Петербург

2021

**Оглавление**

[1 Постановка задачи 3](#_Toc90492568)

[2 Блок-схема программы 4](#_Toc90492569)

[3 Пример работы 5](#_Toc90492570)

[3.1 Функция main, представляющая пример использования написанной программы: 5](#_Toc90492571)

[3.2 Результат выполнения: 6](#_Toc90492572)

[4 Текст программы 7](#_Toc90492573)

[5 Вывод 13](#_Toc90492574)

# Постановка задачи

Целью данной задачи является разработка алгоритма ввода с клавиатуры чисел различных типов данных и отображения их внутреннего представления в двоичной системе счисления. Дополнить разработанный алгоритм блоками для выполнения преобразования полученного двоичного кода исходного типа данных и последующего вывода преобразованного кода в двоичной системе счисления и в формате исходного данного.

Используемые типы данных:

int, long double

Дополнительное преобразование:

Поменять местами заданные пользователем группы рядом стоящих бит, номера старших разрядов этих групп и количество бит в группе, вводится с клавиатуры.

# Блок-схема программы

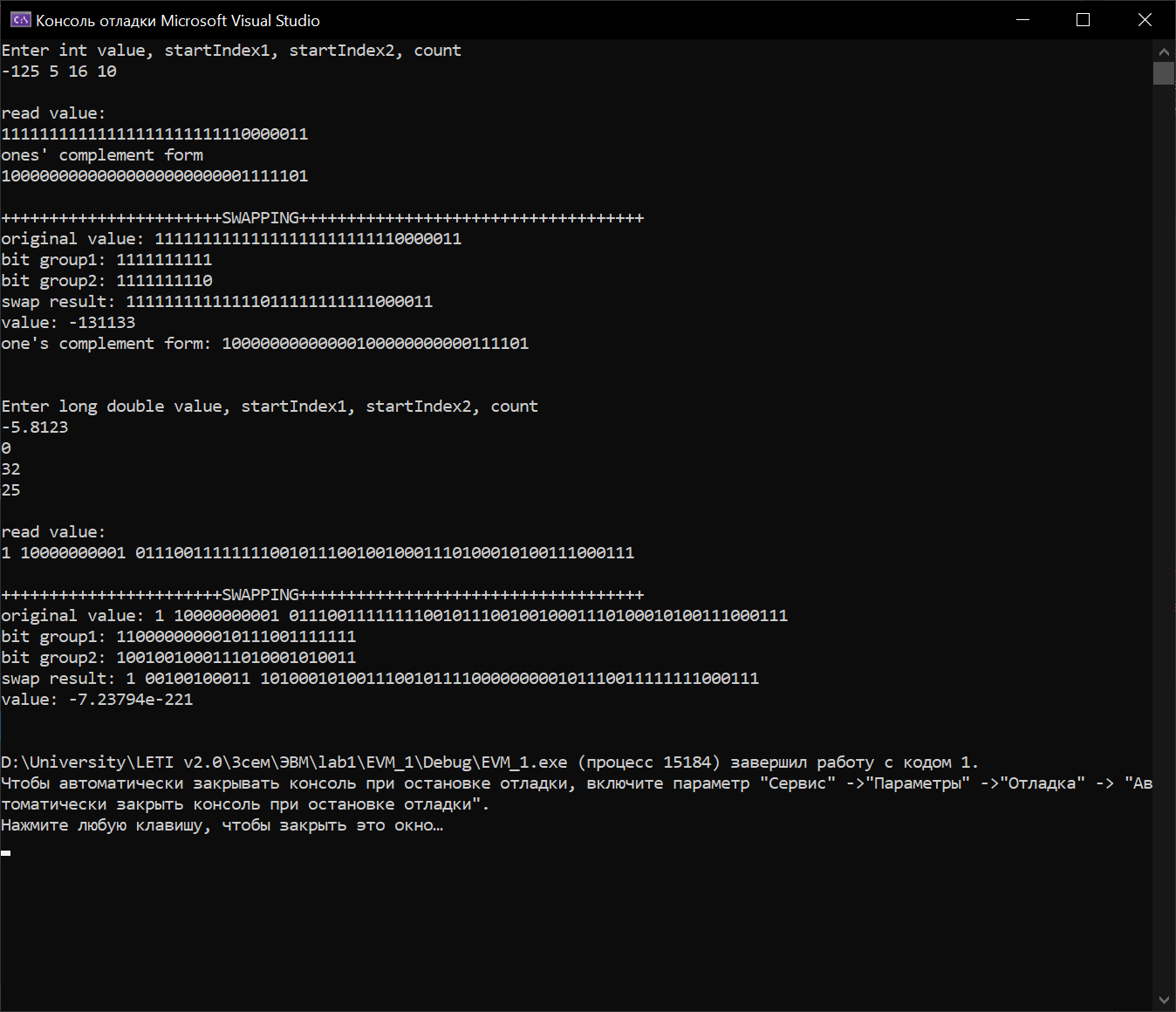


# Пример работы

## Функция main, представляющая пример использования написанной программы:

|  |
| --- |
| using namespace std;  #include <iostream>  #include "BufferValue.h"  #include "IntRipper.h"  #include "LongDoubleRipper.h"  int main()  {  //int, long double  //task 5: swap bit groups  //input vars  int inputInt, startIndex1, startIndex2, count;  long double inputDouble;  //classes for value processing  LongDoubleRipper doubleRipper;  IntRipper intRipper;  ////////////////////////////////////////////////  ///////32 bit int processing.../////////////////  ////////////////////////////////////////////////  cout << "Enter int value, startIndex1, startIndex2, count\n";  //get int input  if (scanf\_s("%i%i%i%i", &inputInt, &startIndex1, &startIndex2, &count) != 4)  {  cout << "Invalid input";  return 0;  }    cout << "\n";  //print int bits  intRipper.PrintBits(inputInt);  //print swap result  cout << "\n+++++++++++++++++++++++SWAPPING++++++++++++++++++++++++++++++++++++\n";  intRipper.SwapBitGroups(inputInt, startIndex1, startIndex2, count);  cout << "\n";  ////////////////////////////////////////////////  ///////64 bit float processing...///////////////  ////////////////////////////////////////////////  cout << "Enter long double value, startIndex1, startIndex2, count\n";  //get input  if (scanf\_s("%lf%i%i%i", &inputDouble, &startIndex1, &startIndex2, &count) != 4)  {  cout << "Invalid input";  return 0;  }  cout << "\n";  //print long double bits  doubleRipper.PrintBits(inputDouble);  //print swap result  cout << "\n+++++++++++++++++++++++SWAPPING++++++++++++++++++++++++++++++++++++\n";  doubleRipper.SwapBitGroups(inputDouble, startIndex1, startIndex2, count);  cout << "\n";  return 1;  } |

## Результат выполнения:



# Текст программы

EVM\_1.cpp

|  |
| --- |
| using namespace std;  #include <iostream>  #include "BufferValue.h"  #include "IntRipper.h"  #include "LongDoubleRipper.h"  int main()  {  //int, long double  //task 5: swap bit groups  //input vars  int inputInt, startIndex1, startIndex2, count;  long double inputDouble;  //classes for value processing  LongDoubleRipper doubleRipper;  IntRipper intRipper;  ////////////////////////////////////////////////  ///////32 bit int processing.../////////////////  ////////////////////////////////////////////////  cout << "Enter int value, startIndex1, startIndex2, count\n";  //get int input  if (scanf\_s("%i%i%i%i", &inputInt, &startIndex1, &startIndex2, &count) != 4)  {  cout << "Invalid input";  return 0;  }    cout << "\n";  //print int bits  intRipper.PrintBits(inputInt);  //print swap result  cout << "\n+++++++++++++++++++++++SWAPPING++++++++++++++++++++++++++++++++++++\n";  intRipper.SwapBitGroups(inputInt, startIndex1, startIndex2, count);  cout << "\n";  ////////////////////////////////////////////////  ///////64 bit float processing...///////////////  ////////////////////////////////////////////////  cout << "Enter long double value, startIndex1, startIndex2, count\n";  //get input  if (scanf\_s("%lf%i%i%i", &inputDouble, &startIndex1, &startIndex2, &count) != 4)  {  cout << "Invalid input";  return 0;  }  cout << "\n";  //print long double bits  doubleRipper.PrintBits(inputDouble);  //print swap result  cout << "\n+++++++++++++++++++++++SWAPPING++++++++++++++++++++++++++++++++++++\n";  doubleRipper.SwapBitGroups(inputDouble, startIndex1, startIndex2, count);  cout << "\n";  return 1;  } |

BufferValue.h

|  |
| --- |
| #pragma once  class BufferValue  {  protected:  int BUFFERSIZE = -1;  virtual bool\* SubBuffer(bool\* buffer, size\_t startIndex, size\_t count)  {  if (startIndex >= BUFFERSIZE || count < 1 || (count + startIndex > BUFFERSIZE))  throw invalid\_argument("Invalid subbuffer args");  bool\* subBuffer = new bool[count];  for (size\_t i = startIndex, c = 0; i < count + startIndex; i++, c++)  subBuffer[c] = buffer[i];  return subBuffer;  }  virtual void WriteToBuffer(bool\* buffer, bool\* subBuffer, size\_t startIndex, size\_t count)  {  for (size\_t i = startIndex, c = 0; i < count + startIndex; i++, c++)  buffer[i] = subBuffer[c];  }  virtual void SwapSubBuffers(bool\* buffer, size\_t startIndex1, size\_t startIndex2, size\_t count)  {  bool\* buffer1 = SubBuffer(buffer, startIndex1, count);  bool\* buffer2 = SubBuffer(buffer, startIndex2, count);  WriteToBuffer(buffer, buffer1, startIndex2, count);  WriteToBuffer(buffer, buffer2, startIndex1, count);  delete[] buffer1, buffer2;  }  virtual void PrintCustomBuffer(bool\* buffer, int count)  {  for (size\_t i = 0; i < count; i++)  cout << buffer[i];  cout << "\n";  }  }; |

IntRipper.h

|  |
| --- |
| #pragma once  class IntRipper : BufferValue  {  public:  IntRipper()  {  BUFFERSIZE = sizeof(int) \* 8; //define int size  }  /// <summary>  /// print integer bits info  /// </summary>  /// <param name="inputInt"></param>  void PrintBits(int inputInt)  {  //create buffer for binary representation  bool\* buffer = new bool[BUFFERSIZE];  ToBuffer(buffer, inputInt);  //print buffer value  cout << "read value:\n";  PrintBuffer(buffer);  //if value < 0 then convert from two's-complement form to one's-complement form  if (inputInt < 0)  {  TwoToOneComplementForm(buffer);  cout << "ones' complement form\n";  PrintBuffer(buffer);  }  delete[] buffer;  }  /// <summary>  /// swaps 2 bit groups within integer  /// </summary>  /// <param name="value">input integer</param>  /// <param name="startIndex1">first bit group start index</param>  /// <param name="startIndex2">second bit group start index</param>  /// <param name="count">size of both bit groups</param>  void SwapBitGroups(int value, int startIndex1, int startIndex2, int count)  {  bool\* buffer = new bool[BUFFERSIZE];  ToBuffer(buffer, value);  cout << "original value: ";  PrintBuffer(buffer);  //get bit groups as arrays  bool\* buffer1 = SubBuffer(buffer, startIndex1, count);  bool\* buffer2 = SubBuffer(buffer, startIndex2, count);  //swap bit groups in main buffer  WriteToBuffer(buffer, buffer1, startIndex2, count);  WriteToBuffer(buffer, buffer2, startIndex1, count);  cout << "bit group1: ";  PrintCustomBuffer(buffer1, count);  cout << "bit group2: ";  PrintCustomBuffer(buffer2, count);  cout << "swap result: ";  PrintBuffer(buffer);  cout << "value: " << FromBuffer(buffer) << "\n";  cout << "one's complement form: ";  TwoToOneComplementForm(buffer);  PrintBuffer(buffer);  cout << "\n";  delete[] buffer, buffer1, buffer2;  }  private:  /// <summary>  /// Reverses negative value buffer  /// </summary>  /// <param name="buf">buffer pointer</param>  void ReverseNegative(bool\* buf)  {  for (int i = 0; i < BUFFERSIZE; i++)  buf[i] = buf[i] == 1 ? 0 : 1;  //make sure first bit is 1  buf[0] = 1;  }  /// <summary>  /// convert buffer from two's-complement form to one's-complement form  /// </summary>  /// <param name="buf">buffer pointer</param>  void TwoToOneComplementForm(bool\* buf)  {  ReverseNegative(buf);  int reversedInt = FromBuffer(buf);  int tobuf = reversedInt + 1;  ToBuffer(buf, tobuf);  }  /// <summary>  /// get int value from buffer contents  /// </summary>  /// <param name="buf">buffer pointer</param>  /// <returns></returns>  int FromBuffer(bool\* buf)  {  int result = 0;  for (size\_t i = 0; i < BUFFERSIZE; i++)  if (buf[i] != 0)  result = result | (1 << (BUFFERSIZE - i - 1));  return result;  }  /// <summary>  /// convert fill buffer with integer bits  /// </summary>  /// <param name="buf">buffer pointer</param>  /// <param name="inputInt">integer to fill from</param>  void ToBuffer(bool\* buf, int inputInt)  {  for (int bitNumber = BUFFERSIZE - 1; bitNumber >= 0; bitNumber--)  {  int compareInt = 1 << bitNumber;  buf[BUFFERSIZE - 1 - bitNumber] = (inputInt & compareInt) != 0 ? 1 : 0;  }  }  void PrintBuffer(bool\* buf)  {  for (size\_t i = 0; i < BUFFERSIZE; i++)  cout << buf[i];  cout << "\n";  }  }; |

LongDoubleRipper.h

|  |
| --- |
| #pragma once  class LongDoubleRipper : BufferValue  {    public:  LongDoubleRipper()  {  BUFFERSIZE = sizeof(long double) \* 8;//define long double size  }  /// <summary>  /// print long double bits info  /// </summary>  /// <param name="inputDouble"></param>  void PrintBits(long double inputDouble)  {  bool\* buffer = new bool[BUFFERSIZE];  ToBuffer(buffer, inputDouble);  cout << "read value:\n";  PrintBuffer(buffer);  delete[] buffer;  }  /// <summary>  /// swaps 2 bit groups within long double  /// </summary>  /// <param name="value">input long double</param>  /// <param name="startIndex1">first bit group start index</param>  /// <param name="startIndex2">second bit group start index</param>  /// <param name="count">size of both bit groups</param>  void SwapBitGroups(long double value, int startIndex1, int startIndex2, int count)  {  bool\* buffer = new bool[BUFFERSIZE];  ToBuffer(buffer, value);  cout << "original value: ";  PrintBuffer(buffer);  //get bit groups as arrays  bool\* buffer1 = SubBuffer(buffer, startIndex1, count);  bool\* buffer2 = SubBuffer(buffer, startIndex2, count);  //swap bit groups in main buffer  WriteToBuffer(buffer, buffer1, startIndex2, count);  WriteToBuffer(buffer, buffer2, startIndex1, count);  cout << "bit group1: ";  PrintCustomBuffer(buffer1, count);  cout << "bit group2: ";  PrintCustomBuffer(buffer2, count);  cout << "swap result: ";  PrintBuffer(buffer);  cout << "value: " << FromBuffer(buffer) << "\n";  delete[] buffer, buffer1, buffer2;  }  private:  /// <summary>  /// get int value from buffer contents  /// </summary>  /// <param name="buf">buffer pointer</param>  /// <returns></returns>  long double FromBuffer(bool\* buf)  {  long long result = 0;  for (size\_t i = 0; i < BUFFERSIZE; i++)  if (buf[i] != 0)  result = result | (1LL << (BUFFERSIZE - i - 1));  LongDoubleUnion u;  u.long\_int = result;  return u.long\_double;  }  /// <summary>  /// convert fill buffer with long double bits  /// </summary>  /// <param name="buf">buffer pointer</param>  /// <param name="inputDouble">long double to fill from</param>  void ToBuffer(bool\* buf, long double inputDouble)  {  LongDoubleUnion u;  u.long\_double = inputDouble;  for (int bitNumber = BUFFERSIZE - 1; bitNumber >= 0; bitNumber--)  {  long long compareInt = 1LL << bitNumber;  buf[BUFFERSIZE - 1 - bitNumber] = (u.long\_int & compareInt) != 0 ? 1 : 0;  }  }  void PrintBuffer(bool\* buf)  {  cout << buf[0];  cout << " ";  for (size\_t i = 1; i < 12; i++)  cout << buf[i];  cout << " ";  for (size\_t i = 12; i < BUFFERSIZE; i++)  cout << buf[i];  cout << "\n";  }  union LongDoubleUnion  {  long double long\_double;  long long long\_int;  };  }; |

# Вывод

В результате выполнения данной работы был реализован алгоритм вывода двоичного представления различных числовых типов данных, а также алгоритм обмена двух групп бит исходного числа с вычислением нового значения числа.