|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Математического обеспечения и стандартизации информационных технологий | |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 1** | |
| **по дисциплине** | |
| **«**Структуры и алгоритмы обработки данных**»**  **Тема: «Поразрядные операции и их применение»** | |
|  | |
| Выполнил студент группы ИКБО-15-22 | Ератин Н. В. |
| Принял преподаватель | Скворцова Л. А. |

|  |  |  |
| --- | --- | --- |
| Самостоятельная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_\_ г. | *(подпись руководителя)* |

Москва 2023

Содержание

[**1.** **Цель работы** 3](#_Toc146257102)

[**2.** **Задача №1** 3](#_Toc146257103)

[**2.1 Постановка задачи** 3](#_Toc146257104)

[**2.2 Реализация решения** 3](#_Toc146257105)

[**2.2.1 Упражнение №1** 3](#_Toc146257106)

[**2.2.1.1 Тестирование Упражнение №1** 4](#_Toc146257107)

[**2.2.2 Упражнение №2** 5](#_Toc146257108)

[**2.2.2.1 Тестирование Упражнение №2** 5](#_Toc146257109)

[**2.2.3 Упражнение №3** 6](#_Toc146257110)

[**2.2.3.1 Тестирование Упражнение №3** 6](#_Toc146257111)

[**2.2.4 Упражнение №4** 6](#_Toc146257112)

[**2.2.4.1 Тестирование Упражнение №4** 7](#_Toc146257113)

[**2.2.5 Упражнение №5** 7](#_Toc146257114)

[**2.2.5.1 Тестирование Упражнение №5** 8](#_Toc146257115)

[**3. Задача №2** 8](#_Toc146257116)

[**3.1 Постановка задачи** 8](#_Toc146257117)

[**3.2 Алгоритм решения** 9](#_Toc146257118)

[**3.3 Описание структуры, используемой в решении, для представления данных в оперативной памяти.** 10](#_Toc146257119)

[**3.4 Тесты** 11](#_Toc146257120)

[**4. Вывод** 15](#_Toc146257121)

# **Цель работы**

Получение навыков применения поразрядных операций в алгоритмах.

# **Задача №1**

## **2.1 Постановка задачи**

Разработать программу, которая продемонстрирует выполнение упражнений варианта. Результаты выполнения упражнения выводить на монитор.

Вариант №11. Условие задания:

|  |  |  |
| --- | --- | --- |
| Упражнение 1 | Номер бита | С 5-ого бита четыре слева |
| Упражнение 2 | Номер бита | 5-ый, 7-ой справа |
| Упражнение 3 | Множитель | 32 |
| Упражнение 4 | Делитель | 32 |
| Упражнение 5 | Задание для выражения | Обнулить n-ый бит, используя маску (вар. 1) |

## **2.2 Реализация решения**

### **2.2.1 Упражнение №1**

Определить переменную целого типа, присвоить ей значение, используя константу в шестнадцатеричной системе счисления. Разработать оператор присваивания и его выражение, которое установит заданные в задании биты исходного значения переменной в значение 1, используя соответствующую маску и поразрядную операцию.

void MyStruct::task1() {

value = 0xA111;

mask = 0xF111;

result = value | mask;

cout << "Value: " << value << "\nBinary: " << toBinary16(value) << endl;

cout << "Result: " << result << "\nBinary: " << toBinary16(mask) << endl;

}

**Код решения Упражнение №1**

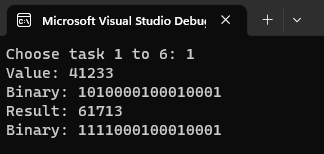
Этот код демонстрирует алгоритм для установки определенных битов в целочисленной переменной value с использованием маски и поразрядной операции "ИЛИ".

**value = 0xA111**; - Объявляется переменная value и ей присваивается значение 0xA111, которое в шестнадцатеричной системе равно двоичному значению 1010000100010001.

**mask = 0xF111;** - Объявляется переменная mask и ей присваивается значение 0xF111, что представляет собой шестнадцатеричное число, равное двоичному значению 1111000100010001.

**value |= mask;** - Выполняется поразрядная операция "ИЛИ" между x и mask.

### **2.2.1.1 Тестирование Упражнение №1**



**Результаты тестирования Упражнение №1**

### **2.2.2 Упражнение №2**

Определить переменную целого типа. Разработать оператор присваивания и его выражение, которое обнуляет заданные в задании биты исходного значения переменной, используя соответствующую маску и поразрядную операцию. Значение в переменную вводится с клавиатуры.

void MyStruct::task2() {

cin >> value;

mask = ~(1 << 4) & ~(1 << 6); // Маска, которая обнуляет пятый и седьмой справа биты

result = value & mask;

cout << "Value: " << value << "\nBinary: " << toBinary8(value) << endl;

cout << "Result: " << result << "\nBinary: " << toBinary8(result) << endl;

}

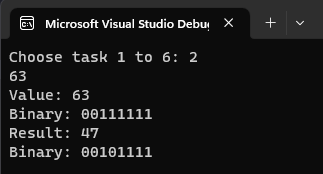
**Код решения Упражнение №2**

**value;** - Объявляется целочисленная переменная value, в которую будет введено значение с клавиатуры.

**mask = ~(1 << 4) & ~(1 << 6);**  - Маска, которая обнуляет пятый и седьмой справа биты.

**result = value & mask;** - Применяется поразрядная операция И между value маской mask. Эта операция, которая обнуляет пятый и седьмой справа биты.

### **2.2.2.1 Тестирование Упражнение №2**



**Результаты тестирования Упражнение №2**

### **2.2.3 Упражнение №3**

Определить переменную целого типа. Разработать оператор присваивания и выражение, которое умножает и делит значение переменной на число, указанное в третьем столбце варианта, используя соответствующую поразрядную операцию. Изменяемое число вводится с клавиатуры.

void MyStruct::task3() {

cin >> value;

result = value << 5;

cout << "Value: " << value << "\nBinary: " << toBinary16(value) << endl;

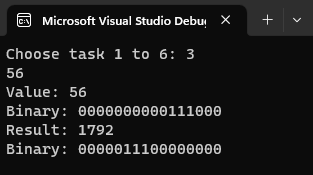
cout << "Result: " << result << "\nBinary: " << toBinary16(result) << endl;

}

**Решение Упражнение №3**

**result = value << 5;** выполняет сдвиг влево на 5 битов, что эквивалентно умножению на 2^5, то есть на 32. Этот сдвиг приводит к увеличению значения result в 2^5 раз, что фактически делает умножение на 32.

### **2.2.3.1 Тестирование Упражнение №3**



**Результаты тестирования Упражнение №3**

### **2.2.4 Упражнение №4**

Определить переменную целого типа. Разработать оператор присваивания и выражение, которое делит значение переменной на число, указанное в третьем столбце варианта, используя соответствующую поразрядную операцию. Изменяемое число вводится с клавиатуры.

void MyStruct::task4() {

cin >> value;

result = value >> 5;

cout << "Value: " << value << "\nBinary: " << toBinary16(value) << endl;

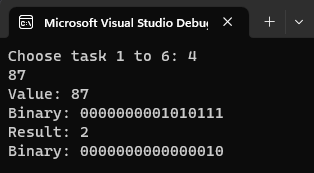
cout << "Result: " << result << "\nBinary: " << toBinary16(result) << endl;

}

**Решение Упражнение №4**

**result = value >> 5;** выполняет сдвиг вправо на 5 битов, что эквивалентно делению на 2^5, то есть на 32. Этот сдвиг приводит к уменьшению значения result в 2^5 раз, что фактически делает деление на 32. Каждый бит, который "выталкивается" за пределы правого конца, теряется.

### **2.2.4.1 Тестирование Упражнение №4**



**Результаты тестирования Упражнение №4**

### **2.2.5 Упражнение №5**

Определить переменную целого типа. Разработать оператор присваивания и выражение, в котором используются только поразрядные операции. В выражении используется маска – переменная. Маска инициализирована единицей в старшем разряде. Изменяемое число вводится с клавиатуры.

void MyStruct::task5() {

cin >> value;

cout << "Enter the bit number to reset (from 0 to 31): ";

cin >> n;

// Создание маски с установленным битом n

mask = 1;

mask = 1 << n;

// Обнуление n-ого бита числа с использованием маски

value = value & (~mask);

cout << "Result: " << value << "\nBinary: " << toBinary16(value) << endl;

cout << "Mask: " << mask << "\nBinary: " << toBinary16(mask) << endl;

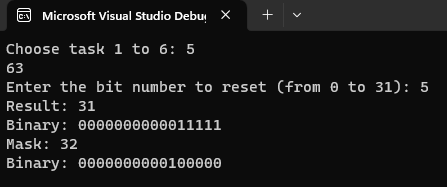
}

**Решение Упражнение №5**

**mask = 1;** - Инициализируется переменная mask значением 1. Это маска, которая имеет только младший бит установленный в 1.

**value = value & (~mask);** - Обнуление n-ого бита числа с использованием маски, приминением обратной маски.

### **2.2.5.1 Тестирование Упражнение №5**



**Результаты тестирования Упражнение №5**

# **3. Задача №2**

## **3.1 Постановка задачи**

Реализовать задачу по сортировке данных файла, используя для представления данных файла (107 семизначных чисел) в памяти, массив битов.

**Дано**: Файл, содержащий не более n (n=107) целых положительных чисел, каждое из которых семизначное число, т.е. принадлежит диапазону [1000000..9999999] и среди них нет повторяющихся.

**Результат**: упорядоченный по возрастанию список чисел

## **3.2 Алгоритм решения**

Для решения данной задачи будет использоваться бит массив. При использовании битового массива для представления сортируемых чисел, программу можно представить как последовательность из трех задач:

1. Инициализация бит массива нулевыми значениями.

2. Считывание целых чисел из файла и установка в 1 соответствующих бит.

3. Формирование упорядоченного выходного файла путем последовательной проверки бит массива и вывод в файл номеров тех бит, которые установлены в 1.

*Листинг 1 – Исходный код программы для Задания 2*

|  |
| --- |
| void MyStruct::task6() {  int size = 0, num\_of\_values;  cout << "Array size: ";  cin >> num\_of\_values;  vector<int> array\_of\_values(num\_of\_values);  for (int i = 0; i < num\_of\_values; i++) {  array\_of\_values[i] = i + 1;  //cout << "Array element " << i << ": ";  //cin >> array\_of\_values[i];  if (array\_of\_values[i] > size) {  size = array\_of\_values[i];  }  }  vector<int> bit\_array(size + 1, 0);  for (int i = 0; i < num\_of\_values; i++) {  bit\_array[array\_of\_values[i]] = 1;  }  cout << "Sorted array: ";  auto start = chrono::high\_resolution\_clock::now();  for (int i = 0; i <= size; i++) {  if (bit\_array[i] == 1) {  cout << i << " ";  }  }  auto end = chrono::high\_resolution\_clock::now(); // Засекаем время после сортировки  chrono::duration<double> duration = end - start; // Вычисляем продолжительность выполнения  cout << "\nTime: " << duration.count() << " sec" << endl;  } |

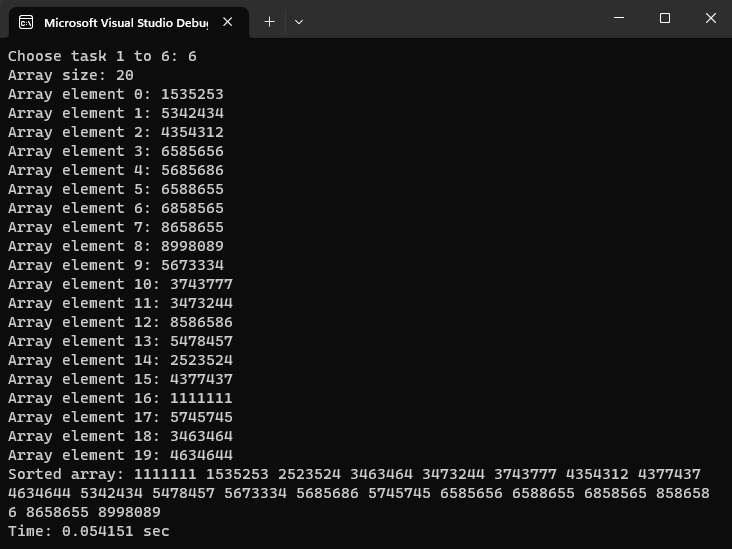
## **3.3 Описание структуры, используемой в решении, для представления данных в оперативной памяти.**

В данной работе используется структура std::bitset - это структура данных в C++, предназначенная для представления фиксированного числа битов в памяти. Она обеспечивает эффективное и удобное управление битами, где каждый бит может быть установлен в 0 или 1. Важные характеристики std::bitset:

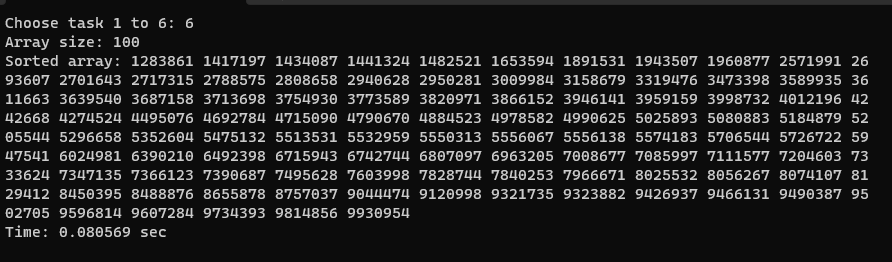
Фиксированный размер: Размер std::bitset фиксирован во время компиляции и не может изменяться во время выполнения программы. Например, std::bitset<8> будет иметь 8 битов.

Поразрядные операции: std::bitset поддерживает поразрядные операции, такие как "И", "ИЛИ", "Исключающее ИЛИ" и др., что позволяет легко выполнять битовые операции.

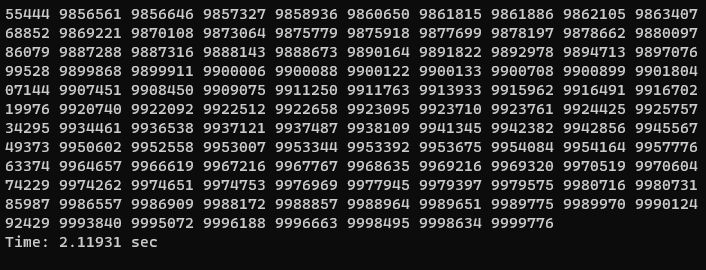
## **3.4 Тесты**

****

**Результат выполнения на 20 значениях**



**Результат выполнения на 100 значениях**



**Результат выполнения на 1000 значениях**

*Листинг 2 – Исходный код проекта*

|  |
| --- |
| #include <chrono>  #include <random>  #include <vector>  #include <iostream>  #include <bitset>  using namespace std;  bitset<16> toBinary16(int value) {  return bitset<16>(value);  }  bitset<8> toBinary8(int value) {  return bitset<8>(value);  }  struct MyStruct  {  public:  int value;  int result;  int mask;  int n;  int i;  void task1();  void task2();  void task3();  void task4();  void task5();  void task6();  };  int main() {  int task;  MyStruct obj;  cout << "Choose task 1 to 6: ";  cin >> task;  switch (task)  {  case 1:  obj.task1();  break;  case 2:  obj.task2();  break;  case 3:  obj.task3();  break;  case 4:  obj.task4();  break;  case 5:  obj.task5();  break;  case 6:  obj.task6();  break;  }  return 0;  }  void MyStruct::task1() {  value = 0xA111;  mask = 0xF111;  result = value | mask; // Применение поразрядной операции ИЛИ для установки битов в значение 1  cout << "Value: " << value << "\nBinary: " << toBinary16(value) << endl;  cout << "Result: " << result << "\nBinary: " << toBinary16(mask) << endl;  }  void MyStruct::task2() {  cin >> value;  mask = ~(1 << 4) & ~(1 << 6); // Маска, которая обнуляет пятый и седьмой справа биты  result = value & mask;  cout << "Value: " << value << "\nBinary: " << toBinary8(value) << endl;  cout << "Result: " << result << "\nBinary: " << toBinary8(result) << endl;  }  void MyStruct::task3() {  cin >> value;  result = value << 5;  cout << "Value: " << value << "\nBinary: " << toBinary16(value) << endl;  cout << "Result: " << result << "\nBinary: " << toBinary16(result) << endl;  }  void MyStruct::task4() {  cin >> value;  result = value >> 5;  cout << "Value: " << value << "\nBinary: " << toBinary16(value) << endl;  cout << "Result: " << result << "\nBinary: " << toBinary16(result) << endl;  }  void MyStruct::task5() {  cin >> value;  cout << "Enter the bit number to reset (from 0 to 31): ";  cin >> n;  // Создание маски с установленным битом n  mask = 1;  mask = 1 << n;  // Обнуление n-ого бита числа с использованием маски  value = value & (~mask);  cout << "Result: " << value << "\nBinary: " << toBinary16(value) << endl;  cout << "Mask: " << mask << "\nBinary: " << toBinary16(mask) << endl;  }  void MyStruct::task6() {  int size = 9999999, num\_of\_values;  cout << "Array size: ";  cin >> num\_of\_values;  vector<int> array\_of\_values(num\_of\_values);  random\_device rd;  default\_random\_engine generator(rd());  uniform\_int\_distribution<int> distribution(1000000, 9999999);  for (int i = 0; i < num\_of\_values; i++) {  array\_of\_values[i] = distribution(generator); // Автоматическое заполнение массива 7-значными случайными числами  }  /\* for (int i = 0; i < num\_of\_values; i++) {  cout << "Array element " << i << ": ";  cin >> array\_of\_values[i];  if (array\_of\_values[i] > size) {  size = array\_of\_values[i];  }  } \*/    vector<int> bit\_array(size + 1, 0);  for (int i = 0; i < num\_of\_values; i++) {  bit\_array[array\_of\_values[i]] = 1;  }  cout << "Sorted array: ";  auto start = chrono::high\_resolution\_clock::now();  for (int i = 0; i <= size; i++) {  if (bit\_array[i] == 1) {  cout << i << " ";  }  }  auto end = chrono::high\_resolution\_clock::now(); // Засекаем время после сортировки  chrono::duration<double> duration = end - start; // Вычисляем продолжительность выполнения  cout << "\nTime: " << duration.count() << " sec" << endl;  } |

# **4. Вывод**

В результате выполнения работы я:

1. Освоил алгоритмы работы с поразрядными операциями и их реализацию на языке программирования C++

2. Научился программировать автоматическое тестирование простых программ