|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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 ИНБО-12-23 | Албахтин И. В. |
| Принял преподаватель | Муравьева Е. А. |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

Москва 2024

**Оглавление**

[1. Цель работы 3](#_Toc176954242)

[2. Задача №1 3](#_Toc176954243)

[2.1 Постановка задачи 3](#_Toc176954244)

[2.2 Реализация решения 3](#_Toc176954245)

[2.2.1 Упражнение №1 3](#_Toc176954246)

[2.2.1.1 Тестирование Упражнение №1 4](#_Toc176954247)

[2.2.2 Упражнение №2 4](#_Toc176954248)

[2.2.2.1 Тестирование Упражнение №2 5](#_Toc176954249)

[2.2.3 Упражнение №3 5](#_Toc176954250)

[2.2.3.1 Тестирование Упражнение №3 6](#_Toc176954251)

[2.2.4 Упражнение №4 6](#_Toc176954252)

[2.2.4.1 Тестирование Упражнение №4 7](#_Toc176954253)

[2.2.5 Упражнение №5 7](#_Toc176954254)

[2.2.5.1 Тестирование Упражнение №5 8](#_Toc176954255)

[3.Задача №2 8](#_Toc176954256)

[3.1 Постановка задачи 8](#_Toc176954257)

[3.2 Алгоритм решения 8](#_Toc176954258)

[3.2.1 Тестирование задачи №2 9](#_Toc176954259)

[3.3 Задача №2.1 10](#_Toc176954260)

[3.3.1 Тестирование задачи №2.1 11](#_Toc176954261)

[4. Задача №3 11](#_Toc176954262)

[4.1 Постановка задачи 11](#_Toc176954263)

[4.2 Алгоритм решения 12](#_Toc176954264)

[4.3 Описание структуры, используемой в решении, для представления данных в оперативной памяти. 13](#_Toc176954265)

[5. Вывод 14](#_Toc176954266)

# **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 №3. Условие задания:

|  |  |  |
| --- | --- | --- |
| Упражнение 1 | Номер бита | Только с четными номерами |
| Упражнение 2 | Номер бита | 7-ой, 9-ый, 11-ый |
| Упражнение 3 | Множитель | 16 |
| Упражнение 4 | Делитель | 16 |
| 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.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Рисунок 1. Код решения Упражнение №1**

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

**Unsigned short int var1 = 0x8C**; - Объявляется переменная var1 и ей присваивается значение **0x8C**, которое в шестнадцатеричной системе равно двоичному значению 0000 0000 1000 1100.

**Unsigned short int mask = 0x5555;** - Объявляется переменная mask и ей присваивается значение **0x5555**, что представляет собой шестнадцатеричное число, равное двоичному значению 0101 0101 0101 0101.

**var1 |= mask;** - Выполняется поразрядная операция "ИЛИ" между var1 и mask.

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

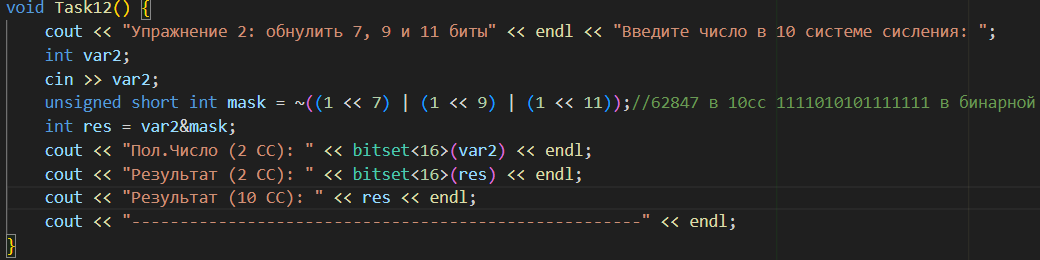
Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Рисунок 2. 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.



**Рисунок 3. Код решения Упражнение №2**

**int var2;** - Объявляется целочисленная переменная var2, в которую будет введено значение с клавиатуры.

**unsigned short int mask = ~((1 << 7) | (1 << 9) | (1 << 11));** - Создается маска mask, которая в шестнадцатеричной системе счисления соответствует бинарной последовательности 1111010101111111. Эта маска обнуляет биты 7-й, 9-й и 11-й.

**var2 &= mask;** - Применяется поразрядная операция И между var2 и инвертированной маской mask.

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

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

**Рисунок 4. Результаты тестирования Упражнение №2**

### **2.2.3 Упражнение №3**

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

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Рисунок 5. Решение Упражнение №3**

**unsigned short int res = number << 4;** выполняет сдвиг влево на 4 бита, что эквивалентно умножению на 2^4, то есть на 16.

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

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Рисунок 6. 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 четвертом столбце варианта, используя соответствующую поразрядную операцию. Изменяемое число вводится с клавиатуры.

Изображение выглядит как снимок экрана, текст, Шрифт

Автоматически созданное описание

**Рисунок 7. Решение Упражнение №4**

**unsigned short int result = number >> 4;;** выполняет сдвиг вправо на 4 бита, что эквивалентно делению на 2^4, то есть на 16.

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

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Рисунок 8. 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.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание

**Рисунок 9. Решение Упражнение №5**

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

**var5 ^ mask;** - Выполняется операция поразрядного XOR между var5 и mask. Эта операция устанавливает n-ый бит переменной number в 0.

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

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Рисунок 10. Результаты тестирования Упражнение №5**

# **3.Задача №2. Упражнение 1.**

# **3.1 Постановка задачи**

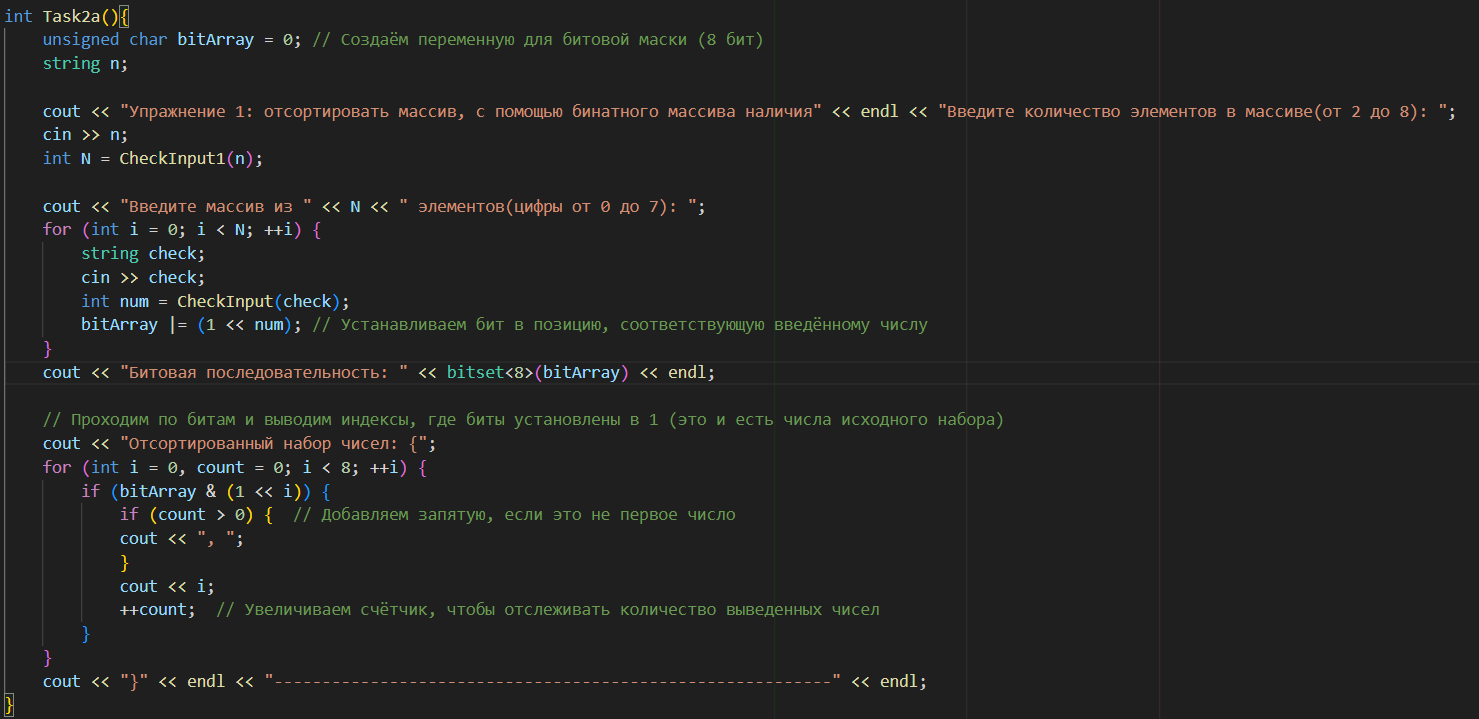
Реализуйте вышеописанный пример с вводом произвольного набора до 8-ми чисел (со значениями от 0 до 7) и его сортировкой битовым массивом в виде числа типа unsigned char. Проверьте работу программы. Исправьте программу задания, чтобы для сортировки набора из 64-х чисел использовалось не одно число типа unsigned long long, а линейный массив чисел типа unsigned char.

## **3.2 Алгоритм решения**

1. Создаем переменную для битовой маски.

2. Далее вводим числа и устанавливаем бит, соответствующий введенному числу.

3. Выводим битовую последовательность.



**Рисунок 11. Решение Задачи №2, упражнение 1.**

### **3.2.1 Тестирование задачи №2, упражнение 1.**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Рисунок 12. Результаты тестирования Задачи №2, упражнения 1.**

## **3.3 Задачи №2, упражнение 2.**

|  |
| --- |
| int CheckInput1 (string n){      while (!(checkNum(n) & (stoi(n) >= 2 & stoi(n) <= 8))) {          cout << "Введено некорректное значение, попробуйте еще раз: ";          cin >> n;      }      int N = stoi(n);      return N;  }  int CheckInput2 (string n){      while (!(checkNum(n) & (stoi(n) >= 0 & stoi(n) <= 63))) {          cout << "Введено некорректное значение, попробуйте еще раз: ";          cin >> n;      }      int N = stoi(n);      return N;  }  void Task2b(){      const int SIZE = 8; // Размер массива для 64 бит (8 байтов)      unsigned char bitArray[SIZE] = { 0 }; // Массив для хранения битов      string n; cout << "Упражнение 2: усовершенствовать работу алгоритма из 1 упражнения" << endl << "Введите количество элементов в массиве(от 2 до 8): "; cin >> n;      int N = CheckInput1(n);        cout << "Введите " << N << " числа от 0 до 63:" << endl;      for (int i = 0; i < N; ++i) {          string checkNum;          cin >> checkNum;          int num = CheckInput2(checkNum);          // Определяем индекс в массиве и позицию бита          int index = num / 8; // Позиция байта          int bitPos = num % 8; // Позиция бита внутри байта          // Устанавливаем бит в позицию, соответствующую введённому числу          bitArray[index] |= (1 << bitPos);      }      // Вывод битовой последовательности      cout << "Битовая последовательность: ";      for (int i = 0; i < SIZE; ++i) {          cout << bitset<8>(bitArray[i]) << " ";      }      // Проходим по битам и выводим индексы, где биты установлены в 1 (это и есть числа исходного набора)      cout << endl << "Отсортированный набор чисел: {";      bool first = true; // Флаг для управления форматированием вывода      for (int i = 0; i < SIZE; ++i) {          for (int j = 0; j < 8; ++j) {              if (bitArray[i] & (1 << j)) {                  if (!first) {  // Добавляем запятую, если это не первое число                      cout << ", ";                  }                  cout << (i \* 8 + j);                  first = false; // После первого числа убираем флаг              }          }      }  }  int main() {      setlocale(LC\_ALL, "RUS");      cout << "Задание 2, упражнение 2" << endl << "--------------------------------------------------------------" << endl;      Task2b();  } |

### **3.3.1 Тестирование задачи №2, упражнение 2.**

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание**

**Рисунок 12. Результаты тестирования задачи №2, упражнение 2.**

# **4. Задача №3**

## **4.1 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, среди них нет повторяющихся.

**Результат**: упорядоченный по возрастанию список чисел, сохраненных в файле.

**Ограничения**:

- Доступной оперативной памяти 1МБ.

- Дисковая память неограниченна.

- Время выполнения программы 10 секунд.

## **4.2 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*

|  |
| --- |
| #include <iostream>  #include <fstream>  #include <bitset>  using namespace::std;  const int MAX\_NUM = 100000;  // Функция для получения текущего объема использованной оперативной памяти (в байтах)  long getMemoryUsage() {      PROCESS\_MEMORY\_COUNTERS memInfo;      GetProcessMemoryInfo(GetCurrentProcess(), &memInfo, sizeof(memInfo));      return memInfo.PeakWorkingSetSize / 1024; // Возвращает максимальный объем памяти в килобайтах  }  const int MAX\_NUM = 10000000;  void boss(const string &inputFilename, const string &outputFilename) {      bitset<MAX\_NUM> bitArray;      ifstream in(inputFilename);      if (!in) {          cerr << "Ошибка открытия файла для чтения!" << endl;          return;      }      int number;      while (in >> number) {          if (number >= 0 && number < MAX\_NUM) {              bitArray.set(number);          }      }      in.close();      ofstream out(outputFilename);      if (!out) {          cerr << "Ошибка открытия файла для записи!" << endl;          return;      }      for (int i = 0; i < MAX\_NUM; ++i) {          if (bitArray.test(i)) {              out << i << "\n";          }      }      out.close();  }  int start(){      string inputFilename, outputFilename;      cout << "Введите название файла с числами: ";      cin >> inputFilename;      cout << "Введите название файла куда будем сохранять отсортированное: ";      cin >> outputFilename;      long memoryBefore = getMemoryUsage();      auto start = chrono::high\_resolution\_clock::now();      // Вызов функции для обработки чисел      boss(inputFilename, outputFilename);      // Измерение времени окончания выполнения программы      auto end = chrono::high\_resolution\_clock::now();      // Вычисление затраченного времени      chrono::duration<double> duration = end - start;      // Измерение объема памяти после выполнения программы      long memoryAfter = getMemoryUsage();      // Вывод времени выполнения и памяти      cout << "Затраченное время: " << duration.count() << " секунд" << endl;      cout << "Использованная память: " << (memoryAfter - memoryBefore) << " KB" << endl;      return 0;  }  int main(){      setlocale(LC\_ALL, "RUS");     cout << endl << "Задание 3" << endl << "--------------------------------------------------------------" << endl;      start();      return 0;  } |

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

В данной работе используется структура 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 bitset фиксирован во время компиляции и не может изменяться во время выполнения программы. Например, std::bitset<8> будет иметь 8 битов.

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

Легкий доступ: Каждый бит в bitset имеет индекс, который можно использовать для доступа к нему. Например, myBitset[3] обратится к третьему биту в myBitset.

Удобное использование bitset часто используется для флагов, битовых масок и других задач, связанных с битами.

# **5. Вывод**

В результате выполнения работы я освоил приёмы работы с битовым представлением беззнаковых целых чисел и способы реализации эффективных алгоритмов сортировки на основе битового массива.