|  |  |  |
| --- | --- | --- |
|  |  |  |
| 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»

по теме «Поразрядные операции»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-13-22 | Гиль Николай Михайлович |
| **Проверил:** | ассистент Муравьёва Е.А. |

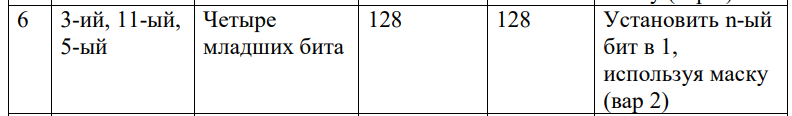
МОСКВА 2023 г.

**Практическая работа № 1**

**Цель работы**

Освоить приёмы работы с битовым представлением беззнаковых целых чисел, реализовать эффективный алгоритм сортировки на основе битового массива.

**Ход работы**

**Вариант 6**

**Задание 1**

**Формулировка задачи:**

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.

**Решение:**

* Для выполнения данной задачи нам вводится число в 16-ой системе, которое переводится в 2-ую 16 битную систему.
* Для установки ***3-его, 11-ого и 5-ого битов*** этого числа в еденицу побитно сложим его с данной маской:(***0000100000101000***) и получим нужное число.
* Полученное число переведем в 16-ую систему.

|  |
| --- |
| unsigned int n;  cout << "Исходное число: ";  cin >> hex >> n;  unsigned int mask = 0b0000100000101000;  cout << std::bitset<16>(n) << "- число" << endl;  cout << std::bitset<16>(mask) << "- маска" << endl;  cout << std::bitset<16>(n | mask) << "- результат в 2-ой" << endl;  cout << "результат:" << hex << (n | mask); |

Листинг 1 – код задачи 1.1

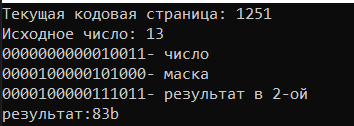


Рисунок 1 – результат тестирования 1.1

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.

**Решение:**

* Для выполнения данной задачи нам вводится число в 16-ой системе, которое переводится в 2-ую 16 битную систему.
* Для установки ***4-ёх младших битов*** этого числа в нули побитно умножим его с данной маской:(***1111111111110000***) и получим нужное число.
* Полученное число переведем в 16-ую систему

|  |
| --- |
| unsigned int n;  cout << "Исходное число: ";  cin >> hex >> n;  unsigned int mask = 0b1111111111110000;  cout << std::bitset<16>(n) << "- число" << endl;  cout << std::bitset<16>(mask) << "- маска" << endl;  cout << std::bitset<16>(n & mask) << "- результат в 2-ой" << endl;  cout << "результат:" << hex << (n & mask); |

Листинг 2 – код задачи 1.2

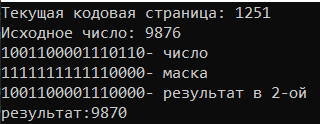


Рисунок 2 – результат тестирования 1.2

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.

**Решение:**

* Для выполнения данной задачи нам вводится число в 16-ой системе, которое переводится в 2-ую 16 битную систему.
* Для увеличения ***в 128 раз*** этого числа произведем побитный сдвиг в лево на 7 (степень 2 для получения числа 128).
* Полученное число переведем в 16-ую систему

|  |
| --- |
| unsigned int n;  cout << "Исходное число: ";  cin >> hex >> n;  cout << std::bitset<16>(n) << "- число" << endl;  n <<= 7;  cout << std::bitset<16>(n) << "- результат в 2-ой" << endl;  cout << "результат:" << hex << (n); |

Листинг 3 – код задачи 1.3

Рисунок 3 – результат тестирования 1.3

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.

**Решение:**

* Для выполнения данной задачи нам вводится число в 16-ой системе, которое переводится в 2-ую 16 битную систему.
* Для уменьшения ***в 128 раз*** этого числа произведем побитный сдвиг в право на 8 (степень 2 для получения числа 256).
* Полученное число переведем в 16-ую систему

|  |
| --- |
| unsigned int n;  cout << "Исходное число: ";  cin >> hex >> n;  cout << std::bitset<16>(n) << "- число" << endl;  n >>= 8;  cout << std::bitset<16>(n) << "- результат в 2-ой" << endl;  cout << "результат:" << hex << (n); |

Листинг 4 – код задачи 1.4

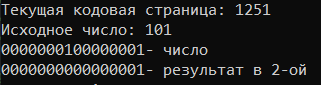


Рисунок 4 – результат тестирования 1.4

5) Определить переменную целого типа. Разработать оператор присваивания и выражение, в котором используются только поразрядные операции. В выражении используется маска – переменная. Маска может быть инициализирована единицей в младшем разряде (вар 1) или единицей в старшем разряде (вар 2). Изменяемое число вводится с клавиатуры.

**Решение:**

* Для выполнения данной задачи нам вводится число в 16-ой системе, которое переводится в 2-ую 16 битную систему и выбирается ***бит для перестановки в единицу***.
* Для установки ***выбранного бита в 1*** сложим его с данной маской:(***1000000000000000***) которую мы сдвинем в право на (15 минус номер выбранного бита) и получим нужное число.
* Полученное число переведем в 16-ую систему

|  |
| --- |
| unsigned int buf;  int n;  cout << "Исходное число: ";  cin >> hex >> buf;  cout << std::bitset<16>(buf) << "- число" << endl;  unsigned short int maska = 0b1000000000000000;  cout << "Выбор бита: ";  cin >> dec >> n;  maska = (maska >> (15 - n));  cout << std::bitset<16>(maska) << "- маска" << endl;  buf = buf | maska;  cout << std::bitset<16>(buf) << "- результат в 2-ой" << endl;  cout << "результат:" << hex << buf; |

Листинг 5 – код задачи 1.5

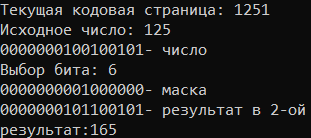


Рисунок 5 – результат тестирования 1.5

**Задание 2**

Формулировка задачи:

1. Реализовать 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.

**Решение:**

* Вводится n чисел, которые сохраняются в векторе. Создается переменная a – хранилище битов.
* Затем устанавливаются соответствующие биты в переменной a в зависимости от введенных чисел.
* На основе этого получаем двоичный вектор, который позволяет по нему отследить какие есть числа и отсортировать массив.

|  |
| --- |
| vector<int> arr;  unsigned char a = 0;  int n, buf;  cout << "Кол-во чисел: ";  cin >> n;  for (int i = 0; i < n; i++) {  cin >> buf;  arr.push\_back(buf);  }  for (int i = 0; i < n; i++) {  a |= (1 << arr[i]);  }  for (int i = 0; i < 8; i++) {  //биты из a  cout << ((a >> i) & 1) << " ";  }  cout << endl;  for (int i = 0; i < 8; i++) {  if (((a >> i) & 1) == 1) {  cout << i << " ";  }  } |

Листинг 6 – код задачи 2.1

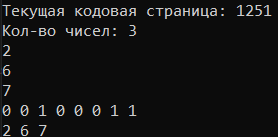


Рисунок 6 – результат тестирования 2.1

1. 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.

**Решение:**

* Логика преобразований таже, но для 64 чисел пришлось расширить ряд из восьми до таблицы 8X8.

|  |
| --- |
| vector<int> arr;  vector<unsigned char> nums(8);  int n, buf;  cout << "Количествово чисел: ";  cin >> n;  for (int i = 0; i < n; i++) {  cin >> buf;  arr.push\_back(buf);  }  for (int x : arr) {  nums[x / 8] |= 1 << (x % 8);  }  for (int i = 0; i < 8; i++) {  for (int j = 0; j < 8; j++) {  cout << ((nums[i] >> j) & 1) << " ";  }  }  cout << endl;  for (int i = 0; i < 64; i++) {  if (((nums[i / 8] >> i % 8) & 1) == 1) {  cout << i << " ";  }  } |

Листинг 7 – код задачи 2.2

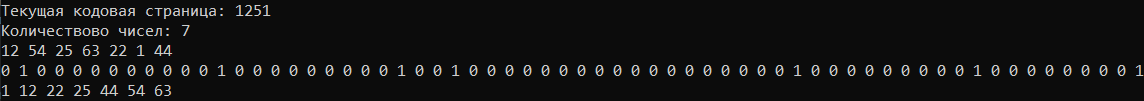


Рисунок 7 – результат тестирования 2.2

**Задание 3**

Формулировка задачи:

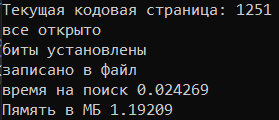
Входные данные: файл, содержащий не более n=107 неотрицательных целых чисел, среди них нет повторяющихся. Результат: упорядоченная по возрастанию последовательность исходных чисел в выходном файле. Время работы программы: ~10 с (до 1 мин. для систем малой вычислительной мощности). Максимально допустимый объём ОЗУ для хранения данных: 1 МБ. Очевидно, что размер входных данных гарантированно превысит 1МБ (это, к примеру, максимально допустимый объём стека вызовов, используемого для статических массивов). Требование по времени накладывает ограничение на количество чтений исходного файла. Реализуйте тестовый пример, демонстрирующий входные данные и заполненный битовый массив (не более 20 чисел). Реализуйте задачу сортировки числового файла для входных данных объемом 100 и 1000 чисел. Показать время выполнения сортировки для каждого объема. Реализуйте задачу сортировки заданного числового файла. В отчёт внесите результаты тестирования для наибольшего количества входных чисел, соответствующего битовому массиву длиной 1МБ и программно определить объём оперативной памяти, занимаемый битовым массивом и время выполнения работы программы (сортировки) для каждого случая.

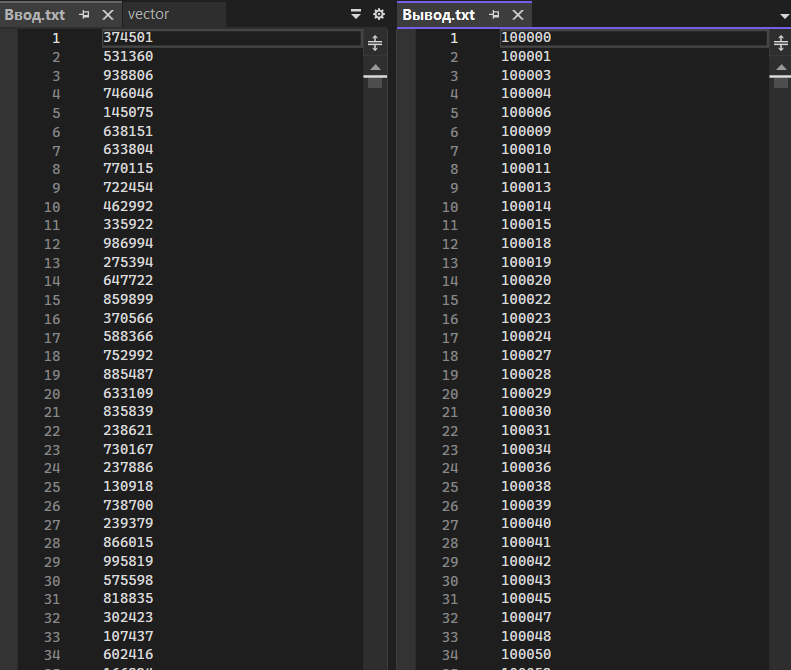
**Решение:**

* Логика преобразований таже, учитывая, что каждое считанное число интерпретируется как индекс бита. Он преобразуется в индекс байта и позицию внутри байта для доступа к нужному биту, и этот бит устанавливается в 1 в векторе nums.

|  |
| --- |
| #include <iostream>  #include <bitset>  #include <vector>  #include <iostream>  #include <fstream>  #include <string>  #include <chrono>  #include<random>  using namespace std;  int Random()  {  random\_device rand\_dev;  mt19937 gen(rand\_dev());  uniform\_int\_distribution<int> random(100000, 999999);  return random(gen);  }  void Task\_1() {  int bytes = (pow(10, 7) + 7) / 8;  vector<unsigned char> nums(bytes);  ofstream f2("Ввод.txt");  for (int i = 0; i < 1000000; i++) {  f2 << Random() << endl;  }  f2.close();  ifstream f("Ввод.txt");  ofstream f1("Вывод.txt");  string line;  cout << "все открыто" << endl;  int bitId, byteInd, setbit;  while (getline(f, line)) {  bitId = stoi(line);  byteInd = bitId / 8; // Индекс байта в котором находится бит  setbit = bitId % 8; // Смещение внутри байта для доступа к биту  nums[byteInd] |= (1 << setbit); // Устанавливаем бит в байте  }  f.close();  cout << "биты установлены" << endl;  auto t0 = chrono::steady\_clock::now();  for (int byteInd = 0; byteInd < bytes; byteInd++) { // запись в файл отсортированных чисел  unsigned char byte = nums[byteInd];  for (int bitId = 0; bitId < 8; bitId++) {  bool isBitSet = (byte & (1 << bitId));  if (isBitSet == 1)  f1 << (byteInd \* 8 + bitId) << endl;  }  }  cout << "записано в файл" << endl;  f1.close();  auto t1 = chrono::steady\_clock::now();  auto fint = chrono::duration\_cast<chrono::microseconds>(t1 - t0);  cout << "время на поиск " << fint.count() / 1000000.0 << endl<<"Пямять в МБ "<<nums.capacity() / (pow(2, 20));  }  int main()  {  setlocale(LC\_ALL, "Russian");  Task\_1();  } |

Листинг 8 – код задачи 3.1





Рисунки 8-9 – результат тестирования для 100 элементов 3.1

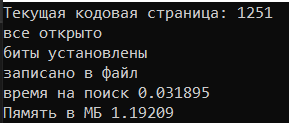


Рисунок 10 – результат тестирования для 1000 элементов 3.1

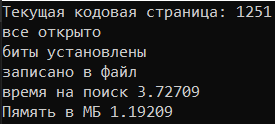


Рисунок 11 – результат тестирования для 8000000 элементов 3.1

**Вывод**

Был практически понят механизм битовой сортировки. Данная сортировка подходит для создания упорядоченных массивов уникальных элементов, так как она не учитывает повторения. Сортировка довольна быстрая, но сложно понять как она работает “под капотом”, хотя базовый принцип совмещения индексации и двоичного переключения состояния интуитивно понятен.