|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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-33-21 | Зарожина Я.А. |
| Принял преподаватель |  |

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

Москва 2022

# **Цель работы**

Получение навыков применения поразрядных операций в алгоритмах.

# **Постановка задачи**

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

Требования к упражнениям:

1. Определить переменную целого типа, присвоить ей значение, используя константу в шестнадцатеричной системе счисления. Разработать функцию, которое установит заданные в задании биты исходного значения переменной в значение 1, используя соответствующую маску и поразрядную операцию.
2. 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.
3. 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.
4. 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.
5. 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.
6. Провести тестирование программы на небольших объемах данных, введенных вручную. Разработанные тесты должны покрывать все случаи входных данных (средний, лучший, худший). Результаты тестирования свести в сводные таблицы.
7. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

Вариант №9. Условие задания:

|  |  |
| --- | --- |
| Упражнение 1 | 3-ий, 11-ый, 5-ый |
| Упражнение 2 | Четыре старших бита |
| Упражнение 3 | 128 |
| Упражнение 4 | 128 |
| Упражнение 5 | Установить n-ый бит в 1, используя маску (вар2) |

# **Решение**

Побитовые операции выполняются над отдельными разрядами или битами чисел. Данные операции производятся только над целыми числами.

Поразрядные операции также проводятся только над соответствующими разрядами целочисленных операндов:

* <<: поразрядный сдвиг влево. Оператор сдвигает каждый бит числа на n разрядов влево. Для целых чисел сдвиг влево на n разрядов при отсутствии переполнения является альтернативой умножению на 2n.
* >>: поразрядный сдвиг вправо. Оператор сдвигает каждый бит числа на n разрядов вправо. Для целых чисел сдвиг вправо на n разрядов является альтернативой целочисленному делению на 2n
* &: поразрядная конъюнкция (операция И или поразрядное умножение). Возвращает 1, если оба из соответствующих разрядов обоих чисел равны 1
* |: поразрядная дизъюнкция (операция ИЛИ или поразрядное сложение). Возвращает 1, если хотя бы один из соответствующих разрядов обоих чисел равен 1
* ^: поразрядное исключающее ИЛИ. Возвращает 1, если только один из соответствующих разрядов обоих чисел равен 1
* ~: поразрядное отрицание или инверсия. Инвертирует все разряды операнда. Если разряд равен 1, то он становится равен 0, а если он равен 0, то он получает значение 1.

Упражнение 1

Определена переменная целого типа, присвоено ей значение, я выбрала константу . Разработать функцию, которое установит заданные в задании биты исходного значения переменной в значение 1, используя маску и поразрядную операцию ИЛИ.

|  |
| --- |
| int Exercise1(int x)  {  x = x | 0x1410;  return x;  } |

Упражнение 2

Разработана функция, которая обнуляет 4 старших бита исходного значения целочисленной переменной, введенной пользователем, используя маску (в зависимости от длинны числа, маска увеличится) и поразрядную операцию И.

|  |
| --- |
| int Exercise2(int x, int n)  {  unsigned maska = (0 << 3);  for (int i = 5; i <= n; i++)//сохранение битов, следующих после 4 главных битов(для числа случайной длинны)  {  maska = maska << 1;  maska = maska | 1;  }  x=x& maska;  return x;  } |

Упражнение 3

Разработана функция, которая умножает значение целочисленной переменной, введенной пользователем, на множитель 128, используя поразрядную операцию сдвига на 7 влево.

|  |
| --- |
| int Exercise3(int x)  {  x = x << 7;  return x;  } |

Упражнение 4

Разработана функция, которая делит значение целочисленной переменной, введенной пользователем, на делитель 128, используя поразрядную операцию сдвига вправо на 7.

|  |
| --- |
| int Exercise4(int x)  {  x = x >> 7;  return x;  } |

Упражнение 5

Разработать функцию, реализующую задание, в которой используются только поразрядные операции. В выражении используется маска – переменная. Маска инициализируется единицей в старшем разряде. Изменяемое число и n вводится с клавиатуры.

|  |
| --- |
| int Exercise5(int x, int n)  {  unsigned maska = 0x80000000;  x = x | (maska >> (31 - n));//маска сдвигается до n-ого бита, тем самым устанавливая только нужный бит  return x;  } |

В интерфейсе пользователь видит приглашение для ввода номера задания или завершения работы программы (цифра 0)(Рис. 1).

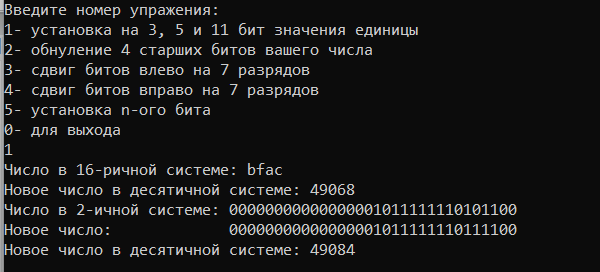


Рис. 1 Интерфейс программы

# **Тестирование**

Упражнение 1:

В первом задании используется число x = (Рис. 2)

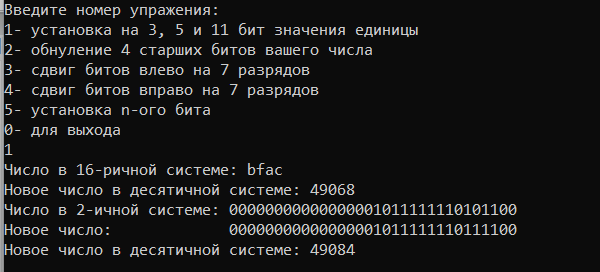


Рис. 2 Первое упражнение

Из результатов выполнения программы видно:

Если в 3 и 5 разряде бит равен 1, а в 11 разряде бит равен 0, то число увеличивается на 16

Таблица 1. Результат тестирования программы для упражнения 1

|  |  |  |
| --- | --- | --- |
| Ввод | Вывод | Ожидаемый результат |
| 49068 | 49084 | 49084 |

Упражнение 2:

Во втором упражнении используется число x = (Рис. 3).

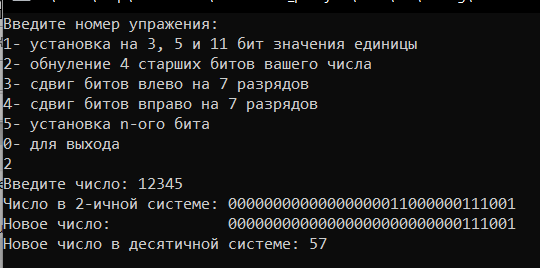


Рис. 3 Второе упражнение

Из результатов выполнения программы видно:

Если обнулить старшие 4 бита числа 12345, то оно уменьшится до 57

Таблица 2. Результат тестирования программы для упражнения 2

|  |  |  |
| --- | --- | --- |
| Ввод | Вывод | Ожидаемый результат |
| 12345 | 57 | 57 |

Упражнение 3:

В задании 3 используется число x = (Рис. 4).

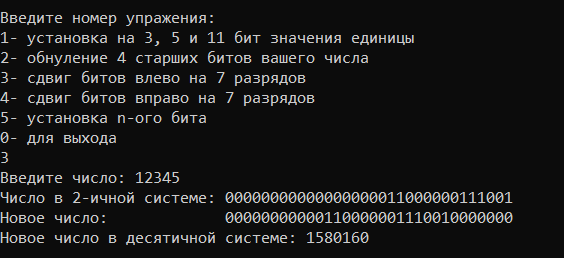


Рис. 4 Третье упражнение

Из результатов выполнения программы видно:

Если число 12345 сдвинуть на 7 бита влево, то оно увеличится в 128 раз

Таблица 3. Результат тестирования программы для упражнения 3

|  |  |  |
| --- | --- | --- |
| Ввод | Вывод | Ожидаемый результат |
| 12345 | 1580160 | 1580160 |

Упражнение 4:

В четвертом задании используется число x = (Рис. 8).

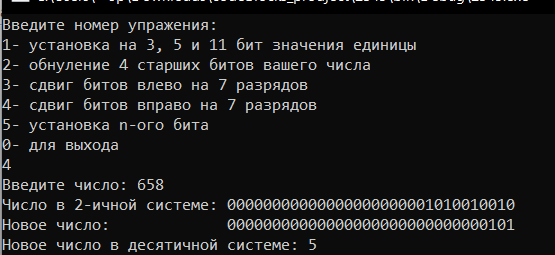


Рис. 5 Четвертое упражнение

Из результатов выполнения программы видно:

Если число 658 сдвинуть на 7 битов вправо, то оно уменьшится в 128 раз

Таблица 4. Результат тестирования программы для упражнения 4

|  |  |  |
| --- | --- | --- |
| Ввод | Вывод | Ожидаемый результат |
| 658 | 5 | 5 |

Упражнение 5:

В пятом задании используется число x = (Рис. 10).

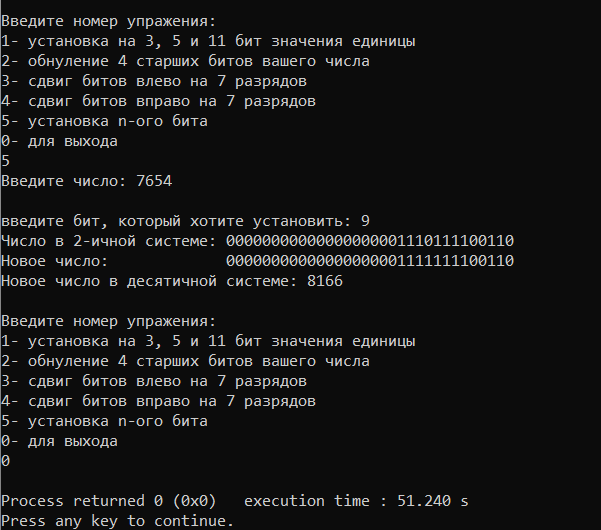


Рис. 6 Пятое упражнение

Из результатов выполнения программы видно:

Если у числа 7654 установится 9 бит, то оно увеличится на 512

Таблица 5. Результат тестирования программы для упражнения 5

|  |  |  |  |
| --- | --- | --- | --- |
| Ввод x | Ввод n | Вывод | Ожидаемый результат |
| 7654 | 9 | 8166 | 8166 |

# **Вывод**

В результате выполнения работы я:

1. Освоила алгоритмы работы с поразрядными операциями и их реализацию на языке программирования C++
2. Научилась программировать автоматическое тестирование простых программ

# **Исходный код программы**

|  |
| --- |
| #include <iostream>  using namespace std;  int Exercise1(int x);  int Exercise2(int x, int n);  int Exercise3(int x);  int Exercise4(int x);  int Exercise5(int x, int n);  void toBin(int x)  {  int n;  n = sizeof(int) \* 8;  unsigned maska = (1 << (n - 1));  for (int i = 1; i <= n; i++)  {  cout << ((x & maska) >> (n - i));  maska = maska >> 1;  }  }  int NBin(int x)  {  int y = 0;  if (x != 0)  {  x = x / 2;  y = y + 1 + NBin(x);  }  return y;  }  int main()  {  setlocale(LC\_ALL, "RUSSIAN");  int number;  cycle:  cout << "Введите номер упражения:\n";  cout << "1- установка на 3, 5 и 11 бит значения единицы \n"<<"2- обнуление 4 старших битов вашего числа\n"  <<"3- сдвиг битов влево на 7 разрядов\n"<<"4- сдвиг битов вправо на 7 разрядов\n"<<"5- установка n-ого бита\n"<<"0- для выхода\n";  cin >> number;  switch (number)  {  case 1:  {  int x = 0xBFAC;  cout << "Число в 16-ричной системе: " << hex << x << "\n";  cout << "Новое число в десятичной системе: " << dec << x << "\n";  cout << "Число в 2-ичной системе: ";  toBin(x);  x = Exercise1(x);  cout << "\nНовое число: ";  toBin(x);  cout << "\nНовое число в десятичной системе: " << dec << x;  cout << "\n\n";  goto cycle;  }  case 2:  {  int x, n;  cout << "Введите число: ";  cin >> x;  n = NBin(x);  cout << "Число в 2-ичной системе: ";  toBin(x);  x = Exercise2(x, n);  cout << "\nНовое число: ";  toBin(x);  cout << "\nНовое число в десятичной системе: " << x;  cout << "\n\n";  goto cycle;  }  case 3:  {  int x;  cout << "Введите число: ";  cin >> x;  cout << "Число в 2-ичной системе: ";  toBin(x);  x = Exercise3(x);  cout << "\nНовое число: ";  toBin(x);  cout << "\nНовое число в десятичной системе: " << x;  cout << "\n\n";  goto cycle;  }  case 4:  {  int x;  cout << "Введите число: ";  cin >> x;  cout << "Число в 2-ичной системе: ";  toBin(x);  x = Exercise4(x);  cout << "\nНовое число: ";  toBin(x);  cout << "\nНовое число в десятичной системе: " << x;  cout << "\n\n";  goto cycle;  }  case 5:  {  int x, n;  cout << "Введите число: ";  cin >> x;  cout << "\nвведите бит, который хотите обнулить: ";  cin >> n;  cout << "Число в 2-ичной системе: ";  toBin(x);  x = Exercise5(x, n);  cout << "\nНовое число: ";  toBin(x);  cout << "\nНовое число в десятичной системе: " << x;  cout << "\n\n";  goto cycle;  }  default:  {  break;  }  }  return 0;  }  int Exercise1(int x)  {  x = x | 0x1410;  return x;  }  int Exercise2(int x, int n)  {  unsigned maska = (0 << 3);  for (int i = 5; i <= n; i++)//сохранение битов, следующих после 4 главных битов(для числа случайной длинны)  {  maska = maska << 1;  maska = maska | 1;  }  x=x& maska;  return x;  }  int Exercise3(int x)  {  x = x << 7;  return x;  }  int Exercise4(int x)  {  x = x >> 7;  return x;  }  int Exercise5(int x, int n)  {  unsigned maska = 0x80000000;  x = x | (maska >> (31 - n));//маска сдвигается до n-ого бита, тем самым устанавливая только нужный бит  return x;  } |

Вывод: Получила навыки применения поразрядных операций в алгоритмах.

Практическая работа № 2

Пример процесса разработки программы с выбором структуры представления данных для оптимизации сортировки.

# **Цель работы**

Рассмотреть пример, демонстрирующий выбор структуры данных, для обеспечения всех требований предъявляемых к разрабатываемой программе.

# **Постановка задачи**

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

Требования к упражнениям:

1. Программа должна сортировать по возрастанию записи файла приблизительно раз в час, т.е. время сортировки должно быть минимальным и составлять секунд десять.
2. 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.
3. Результат. Упорядоченный по возрастанию список чисел, сохраненных в файле.
4. Ограничения:

* Доступной оперативной памяти 1МБ.
* Дисковая память неограниченна.
* Время выполнения программы 10 секунд.

# **Формулировка общего задания**

Проследить процесс разработки алгоритма по заданным условиям. Выполнить задание с использованием оперативной памяти. Реализовать тестовую программу и поставленную с семизначными числами.

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

Примечание. Задача и алгоритм ее решения рассмотрены на практическом занятии. Файл не использовать, смоделируйте тест на последовательности, вводимой с клавиатуры и генерации семизначных чисел.

## **Решение**

Имена объектов алгоритма:

n – количество битов в массиве. bit[n] – битовый массив

**Абстрактное описание алгоритма**

Алгоритм на псевдокоде

//Часть 1: инициализация массива битов нулями

for i←0 t n

bit[i]:=0;

//Часть 2: Заполнение битового массива значениями Чтение числа из файла в переменную i

bit[i]:=1

//Часть 3: Формирование упорядоченного выходного файла

for i←0 t n

if bit[i]=1 then записать i в файл;

## **Неформальное описание алгоритмов трех основных задач реализующих задачу**

Программист (назовем его *Первым программистом*) должен был разработать программу сортировки большого объема данных, хранящихся в файле. Программа нужна через сутки. При выборе алгоритма сортировки он решил посоветоваться с более опытным программистом (назовем его *Вторым программистом*). И у них состоялся диалог, в котором *Второй программист* уточнял особенности решения задачи.

Первое предложение *Второго программиста* – это использовать алгоритм сортировки слиянием, который применяется для сортировки данных в файлах.

Но *Первому программисту* не хотелось углубляться в теорию алгоритмов.

Тогда *Второй программист* предложил использовать готовую программу, опубликованную в литературе по программированию, эта программа имеет 200 строк кода, разбитого на 12 функций. На написание и отладку такой программы *Первый программист* потратил бы не больше недели. Но *Первый программист* сомневался, что проблема будет решена.

Тогда *Второй* *программист* понял, что существуют какие-то требования к программе, он стал расспрашивать *Первого программиста* о его задаче.

В результате *Второй программист* выяснил следующее:

* Разрабатываемая программа будет встроена в большую программу, поэтому она не может использовать системные средства.
* Файл содержит 107 записей. Записи – это целые положительные семизначные числа. Узнав размер файла, *Второй программист* сразу предложил отсортировать записи в оперативной памяти, так как их не так много. Но, *Первый программист* ответил, что хотя компьютер оснащен большим объемом памяти, программа сортировки является лишь малой частью большой программы. К моменту ее выполнения будет свободным лишь 1Мб памяти.
* Программа должна сортировать по возрастанию записи файла приблизительно раз в час, т.е. время сортировки должно быть минимальным и составлять секунд десять.
* Семизначные числа – это номера телефонов.

В результате обсуждения программисты сформулировали точную постановку задачи.

1. **Точная постановка задачи**

**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, сохраненных в файле.

**Ограничения:**

* Доступной оперативной памяти 1МБ.
* Дисковая память неограниченна.
* Время выполнения программы 10 секунд.

1. **Исследование разработки**

Рассмотрение возможных вариантов решения.

**Первый вариант** – использовать однопроходный алгоритм.

Использовать программу сортировки слиянием из 200 строк кода, оптимизировав ее с учетом того, что сортируются семизначные числа. Это сократит размер кода до нескольких десятков строк. Но на кодирование и отладку потребуется несколько дней. Сортировка слиянием для файлов требует использование двух временных файлов. Сортировка слиянием считывает входной файл один раз и разливает его по нескольким временным файлам (два файла), сортирует его с использованием временных файлов, которые считываются и записываются много раз. Выполнение операций чтения и записи во временные файлы осуществляется за длительное время, так как внешняя память имеет низкое быстродействие выполнения операций.

**Второй вариант** – использовать сорока пароходный алгоритм.

Данный подход опирается на использование деталей конкретной задачи.

а) Если представить числа как последовательности символов, то на каждое число потребуется семь байт. Тогда в доступном 1Мб памяти поместится всего 143 000 чисел.

б) Если для представления чисел использовать ячейки из 32 бит(типа int), то в 1Мб поместится уже 250 000 номеров. Тогда программа может сортировать записи файла в оперативной памяти, считывая его за 40 раз (40\*250 000=107): т.е. за первое считывание считываются записи с номерами с 0 по 249 999, которые затем сортируются и записываются в выходной файл. За второй проход сортируются записи с номерами 250 000 по 499 999 и т.д. Для сортировки записей можно использовать алгоритм быстрой сортировки (алгоритм Хоара), который занимает всего 20 строк кода. Вся программа тогда будет содержать около 50 строк кода.

**Третий вариант -** битовая реализация

Оптимальной была бы программа, работающая по схеме, соединяющей преимущества предыдущих: входной файл, считывается один раз и временные файлы не используются.

Такую схему можно реализовать, когда все данные при сортировке будут размещены в 1Мб оперативной памяти. Т.о. задача сводится к отображению 107 целых семизначных чисел приблизительно в восемь миллионов доступных бит (т.к. 1Мб=1024\*1024\*8 бит=8 388 608 бит, а надо 10 000 000 битов, т.е. недостает еще около 2 000 000 битов).

1. **Неформальный алгоритм**

Использование последовательности бит для представления чисел напрашивается само собой. Суть такого представления в следующем.

Пусть имеется 20 целых чисел и их значения в диапазоне от 1 до 20, их надо представить 20 битами. Пусть последовательность чисел, вставляемая в битовую последовательность из 20 бит, будет такой {1, 2, 5, 9, 12, 8}.

Сначала вся последовательность бит (массив длиной 20 бит) заполнена нулями. Биты в последовательности нумеруются с нуля.

Теперь создадим отображение, которое вставляет 1 в бит с номером, равным значению вставляемого числа, тогда получим следующую битовую последовательность: 011001001100100000000.

В решаемой задаче файл представим как строку из десяти миллионов битов (программист нашел недостающие 2 000 000 битов), в которой i-ый бит равен 1, если в файле присутствует число со значением i.

Такой подход возможен в задаче именно с такими исходными данными: числа целые и нет повторяющихся, и их сравнительно не много, чтобы все представить битами в памяти.

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.

ОПИСАНИЕ МОДЕЛИ РЕШЕНИЯ

Описание прототипов алгоритма работы программы.

## Функция Puzir()

Предусловие:

Выбираем способ ввода массива – вручную или автоматически

cout<<"Выберите, как хотите ввести массив:\n0- Вручную\n1- Автоматически\n";

cin>>b;

Вводим элемент массива

for (i = 0; i < n; i++)

{

if (!b)

{

cout << "\nВведите " << i + 1 << " элемент массива ";

cin >> a[i];

}

else

{

a[i] = rand() % 1000;

cout << a[i]<<" ";

}

}

Постусловие:

Программа выводит массив cout << "Массив: ";

Программа выводит отсортированный массив cout << endl<<"Отсортированный массив: ";

Программа выводит время cout << "\n" << "Время: " << elapsed\_ms.count() << " миллисекунд";

Программа выводит cout << "\n Cф+Mф= " << count+move;

cout << endl<<"Отсортированный массив: ";

for (i = 0; i < n; i++)

cout << a[i]<<" ";

cout << "\n" << "Время: " << elapsed\_ms.count() << " миллисекунд";

cout << "\n Cф+Mф= " << count+move;

1. Функция NewMetod ()

Постусловие:

Программа выводит массив cout << "Массив: ";

Программа выводит отсортированный массив cout << endl<<"Отсортированный массив: ";

Программа выводит время cout << "\n" << "Время: " << elapsed\_ms.count() << " миллисекунд";

1. Основная функция main:

Предусловие:

Выбираем метод сортировки – пузырьковый метод или метод вставками

cout << "Выберите метод сортировки:\n1- Пузырьковый метод\n2- Новый метод\n";

cin >> k;

if (k == 1)

Puzir();

if (k == 2)

NewMetod ();

Постусловие:

## Нет

КОДИРОВАНИЕ

# **Исходный код программы**

|  |
| --- |
| #include <iostream>  #include <chrono>  #include <time.h>  #include <ctime>  #include <stdlib.h>  using namespace std;  //сортировка пузырьком  void Puzir()  {  const int n = 5;  int i, k;  int a[n];  unsigned int start\_time, end\_time, search\_time;  start\_time = clock();  bool b;  cout<<"Выберите, как хотите ввести массив:\n0- Вручную\n1- Автоматически\n";  cin>>b;  cout << "Массив: ";  for (i = 0; i < n; i++)  {  if (!b)  {  cout << "\nВведите " << i + 1 << " элемент массива ";  cin >> a[i];  }  else  {  a[i] = rand() % 10000000;  cout << a[i]<<" ";  }  }  end\_time = clock();  search\_time = end\_time - start\_time;  cout << endl<<"Отсортированный массив: ";  for (i = 0; i < n; i++)  cout << a[i]<<" ";  cout<<"\n" << "Время: "<<search\_time<<" миллисекунд" << endl;  }  //сортировка новым способом  void NewMetod()  {  srand(time(NULL));  int i, j, k, z;  bool\* a = new bool[10000000];  int n = 5;  unsigned int start\_time, end\_time, search\_time;  start\_time = clock();  long int s;  cout << endl << "Массив:\n";  for (i = 0; i < n; i++)  {  s = rand() % 10000000;  cout<<s<<endl;  a[s] = true;  }  end\_time = clock();  search\_time = end\_time - start\_time;  cout << endl << "Отсортированный массив:\n";  for(long int i=0;i<1000000;i++)  {  if(a[i] == true)  {  cout<<i<<endl;  }  }  end\_time = clock();  search\_time = end\_time - start\_time;  cout<<"\n" << "Время: "<<search\_time<<" миллисекунд" << endl;  }  int main()  {  setlocale(LC\_ALL, "RUSSIAN");  unsigned short int k;  cout << "Выберите метод сортировки:\n1- Пузырьковый метод\n2- Новый метод\n0- Завершение программы\n";  cin >> k;  if (k == 1)  Puzir();  if (k == 2)  NewMetod();  return 0;  } |

РЕЗУЛЬТАТЫ ТЕСТИРОВНИЯ

В интерфейсе пользователь видит варианты выбора для ввода номера сортировки или завершения работы программы (цифра 0)(Рис. 1).

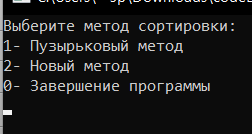


Рис. 1 Интерфейс программы

При вызове пузырькового метода имеем следующий результат выполнения программы:

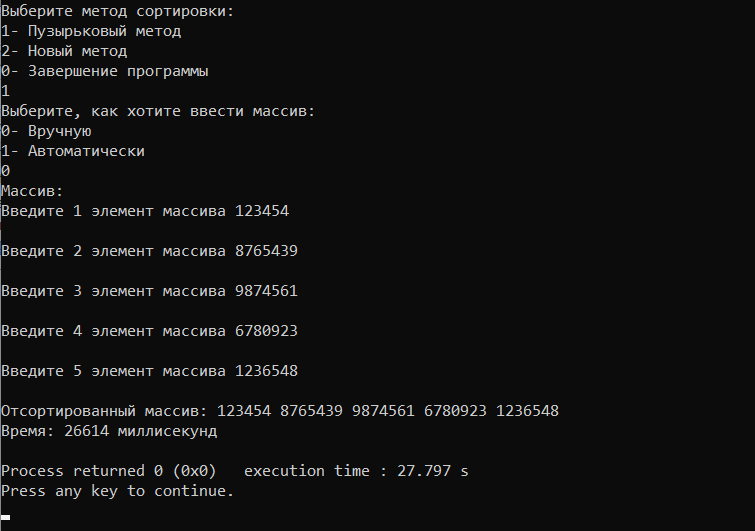


Рис. 2 Пузырьковый метод – вручную

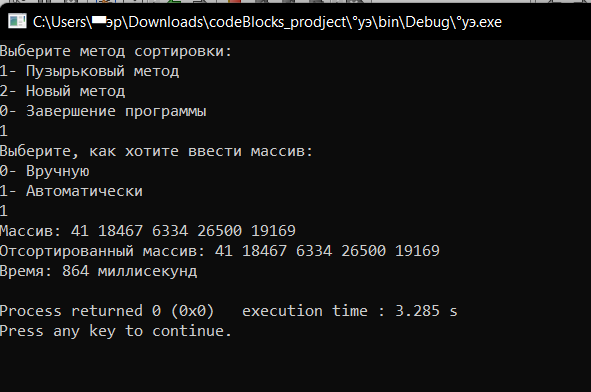


Рис. 3 Пузырьковый метод – автоматически

При вызове нового метода имеем следующий результат выполнения программы:

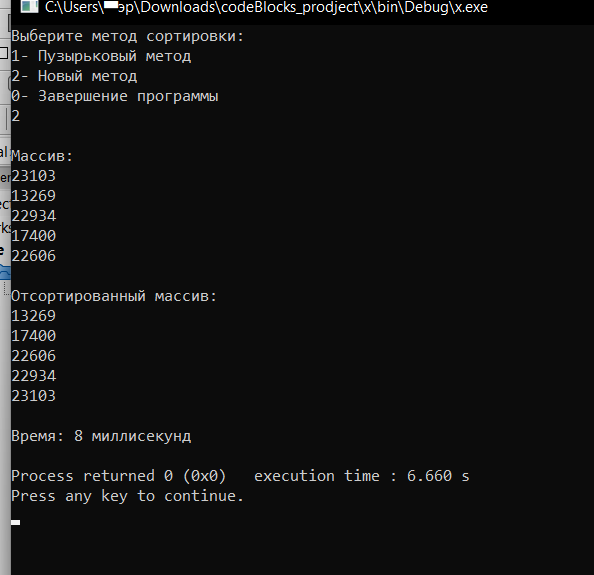


Рис.4 Новый метод - автоматически

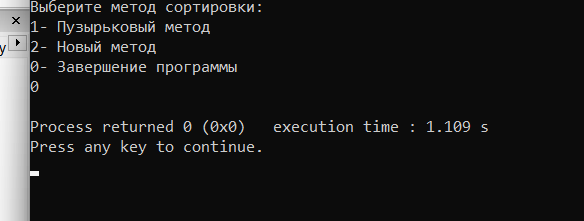


Рис. 5 Выход из программы

ВЫВОД

Внимательное изучила задачи, что позволило найти правильный подход к решению.

Битовые структуры позволили экономно использовать память. Битовые структуры используются для представления плотного набора элементов конечного множества, в который элемент может входить не более одного раза, и, где с элементами не связано более ни каких дополнительных данных.

Использование битового массива привело к сокращению времени выполнения программы. Это произошло за счет того меньшее количество данных требует меньшего времени для обработки, а во-вторых, хранение данных в оперативной памяти, а не на диске сокращает затраты на обращение к ним.

Программа получилась достаточно простой. Простота влечет надежность и эффективность программы, их проще создавать и сопровождать.

При разработке алгоритма использовался подход, основанный на следующих принципах:

1. Понимание предложенной задачи
2. Постановка абстрактной задачи
3. Неформальное описание программы

# СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

Лекция по структуре и алгоритмам обработки данных (Макеева О.В.) от 02.09.22 – лекция в А-18, в 18:00-19:30