|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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-08-22 | Сенькевич Г.С. |
| Принял преподаватель | Филатов А.С. |

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

Москва 2023

# **Цели работы**

* приобретение навыков по определению одномерного массива для структуры данных задачи
* приобретение навыков создания алгоритмов операций над одномерным массивом
* получение навыков по реализации алгоритмов операций над массивом через аппарат функций

# **Постановка задачи**

1. Разработать программу для выполнения операций на статическом массиве
2. Разработать функции для ввода и вывода значений массива.
3. Выполнить декомпозицию задач, определенных вариантом индивидуального задания.
4. Разработать алгоритмы решения задач, определенных вариантом индивидуального задания.
5. Реализовать функции, выполняющие задачи, определенные вариантом индивидуального задания.
6. Разработать программу, демонстрирующую работу всех функций.
7. Изменить разработанную программу так, чтобы все операции выполнялись над динамическим массивом.
8. Изменить разработанную программу так, чтобы все операции выполнялись над динамическим массивом, используя контейнер <vector> для его представления в программе.
9. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

Вариант №23. Условие задания:

|  |  |
| --- | --- |
| Упражнение 1 | Определить, упорядочены ли значения в массиве по возрастанию. |
| Упражнение 2 | Если значения в массиве упорядочены по возрастанию, то удалить из массива элементы, которые кратны введенному значению. |
| Упражнение 3 | Если значения в массиве не упорядочены по возрастанию, то вставить новый элемент в массив перед первым элементом. |

# **Решение**

Массив – это линейная, однородная структура данных, состоящая из конечного числа элементов. Доступ к элементу массива прямой по индексу.

Статический массив – это структура данных, которая создается во время компиляции (т.е. массиву выделяется память). Размер массива нельзя изменять во время работы программы.

Для всех элементов массива общим является имя, но у каждого элемента массива уникальным является индекс.

Имя массива — это константный указатель. Он хранит адрес первого байта области памяти, выделенной под элементы массива.

Доступ к элементу массива осуществляется по имени. Имя элемента массива формируется по следующему правилу:

имя массива [индекс элемента]

Каждый элемент занимает в памяти отдельную ячейку памяти. Ячейка памяти выделяется переменной, таким образом элемент массива – это отдельная переменная в составе массива.

Операции можно выполнять только над элементом массива, допустимые базовым типом массива.

Динамический массив – это структура данных, которая создается во время выполнения программы. В процессе работы программы размер массива может быть изменен.

Динамический массив в языке Си определяется через указатель. Количество элементов в массиве может быть определено перед созданием массива.

Индексация элементов динамического массива начинается с нуля.

# Контейнер <vector> - современная реализация динамических массивов

# Векторы - это реализация динамического массива в C++, т.е. такого массива, чья длина меняется в ходе программы. Называется вектор, потому что добавлять и удалять элементы можно только с одного конца.

# Стандартный шаблон обобщённого программирования языка C++ std::vector<T> — реализация [динамического массива](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D0%BD%D0%B0%D0%BC%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2).

# Типичная реализация вектора — это указатель на динамический массив. Размер вектора — это фактическое число элементов, а объём — количество используемой им памяти.

# Если при вставке в вектор новых элементов, его размер становится больше его объёма, происходит перераспределение памяти. Как правило, это приводит к тому, что вектор выделяет новую область хранения, перемещая элементы и свободные старые области в новый участок памяти.

Для решения первого упражнения была написана функция is\_sorted, проверяющая, упорядочен ли массив по возрастанию. Функция принимает на вход массив и его размер (в случае статического и динамического массива) и возвращает булевую переменную – результат проверки.

|  |
| --- |
| // Функция для статических и динамических массивов  bool is\_sorted(int array[], int size=STATIC\_ARRAY\_SIZE)  {  for (int i = 1; i < size; i++)  {  if (array[i] < array[i - 1])  {  return false;  }  }  return true;  }  // Функция для std::vector  bool is\_sorted(const std::vector<int>& array)  {  for (int i = 1; i < array.size(); i++)  {  if (array[i] < array[i - 1])  {  return false;  }  }  return true;  } |

Для решения второго упражнения была написана функция delete\_multiples, которая удаляет все кратные заданному числу числа в массиве. Функция принимает на вход массив, его размер (в случае статического и динамического массива), число, все кратные которому должны быть удалены из массива, и возвращает целочисленную переменную – количество удалённых из массива элементов.

|  |
| --- |
| // Функция для статических  int delete\_multiples(int array[], int num)  {  int deleted = 0;  for (int i = 0; i < STATIC\_ARRAY\_SIZE - deleted; i++)  {  if (array[i] % num == 0)  {  for (int j = i; j < STATIC\_ARRAY\_SIZE - 1; j++)  {  array[j] = array[j + 1];  }  if (STATIC\_ARRAY\_SIZE > 0)  {  array[STATIC\_ARRAY\_SIZE - 1] = 0;  }  deleted++;  i--;  }  }  return deleted;  }  // Функция для динамических массивов  int delete\_multiples(int\*& array, int size, int num)  {  int deleted = 0;  for (int i = 0; i < size - deleted; i++)  {  if (array[i] % num == 0)  {  for (int j = i; j < size - 1; j++)  {  array[j] = array[j + 1];  }  if (size > 0)  {  array[size - 1] = 0;  }  deleted++;  i--;  }  }  int new\_size = size - deleted;  realloc(array, sizeof(int) \* new\_size);  return deleted;  }  // Функция для std::vector  void delete\_multiples(std::vector<int>& array, int num)  {  for (int i = 0; i < array.size(); i++)  {  if (array[i] % num == 0)  {  array.erase(array.begin() + i);  i--;  }  }  } |

Для решения третьего упражнения была написана функция insert\_first, вставляющая элемент в начал массива. Функция принимает на вход массив, его размер (в случае статического и динамического массива), число, которое должно быть вставлено в начало массива и возвращает целочисленную переменную – количество вставленных в массив элементов.

|  |
| --- |
| // Функция для статических массивов  int insert\_first(int array[], int num)  {  for (int i = STATIC\_ARRAY\_SIZE - 1; i > 0; i--)  {  array[i] = array[i - 1];  }  if (STATIC\_ARRAY\_SIZE > 0)  {  array[0] = num;  return 1;  }  return 0;  }  // Функция для динамических массивов  int insert\_first(int\*& array, int size, int num)  {  realloc(array, sizeof(int) \* (size + 1));  for (int i = size; i > 0; i--)  {  array[i] = array[i - 1];  }  array[0] = num;  return 1;  }  // Функция для std::vector  void insert\_first(std::vector<int>& array, int num)  {  array.push\_back(0);  for (int i = array.size() - 1; i > 0; i--)  {  array[i] = array[i - 1];  }  array[0] = num;  } |

При запуске программы пользователь видит пользовательское меню, позволяющее выбрать между разными типами массивов.

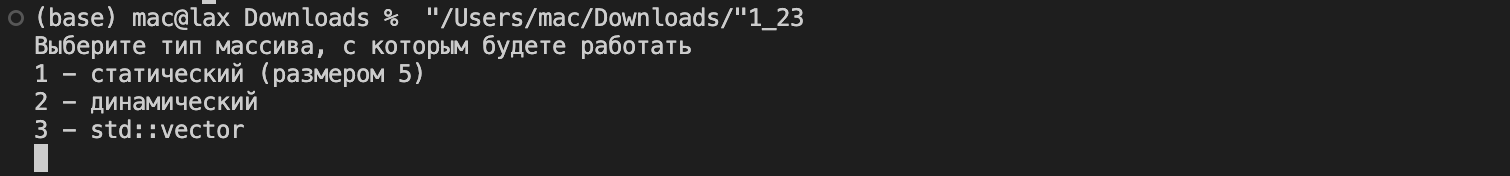


Рисунок 1. Интерфейс программы

# **Тестирование**

Протестируем выполнение программой первого упражнения. Для этого поочередно будем выбирать в меню все возможные типы массива, выберем первую функцию, и введем через пробел значения 12 0 3. Результатом должно быть сообщение о том, что массив не упорядочен по возрастанию. На рисунках 2, 3, 4 видим, как программа вывела верный ответ.

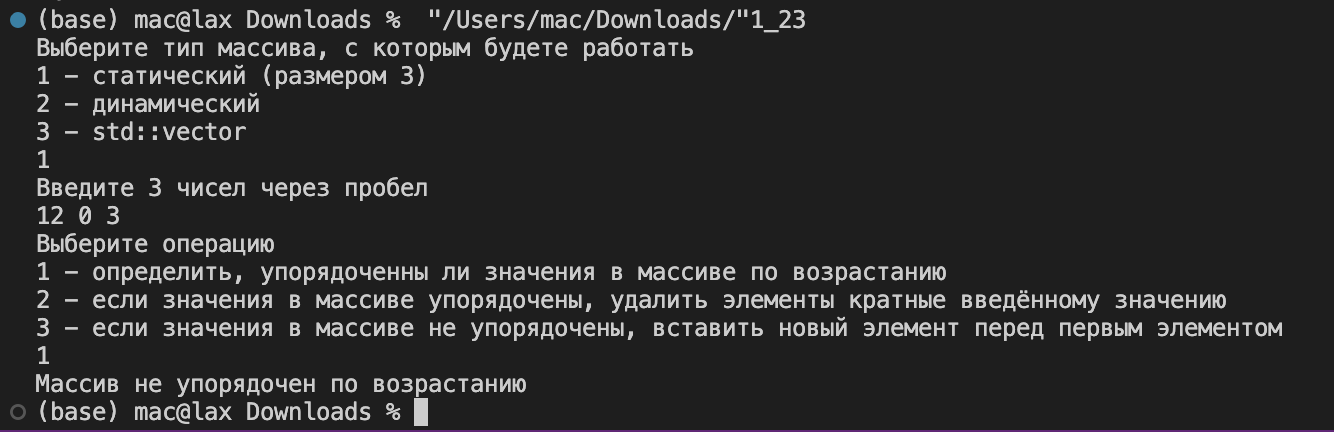
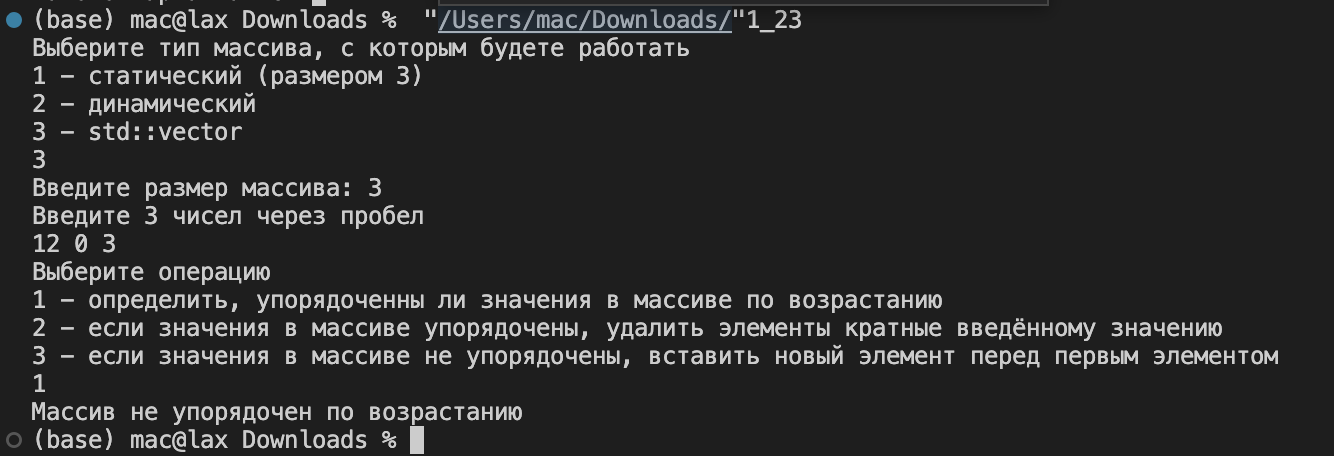


Рисунок 2. Тестирование программы

Рисунок 3. Тестирование программы

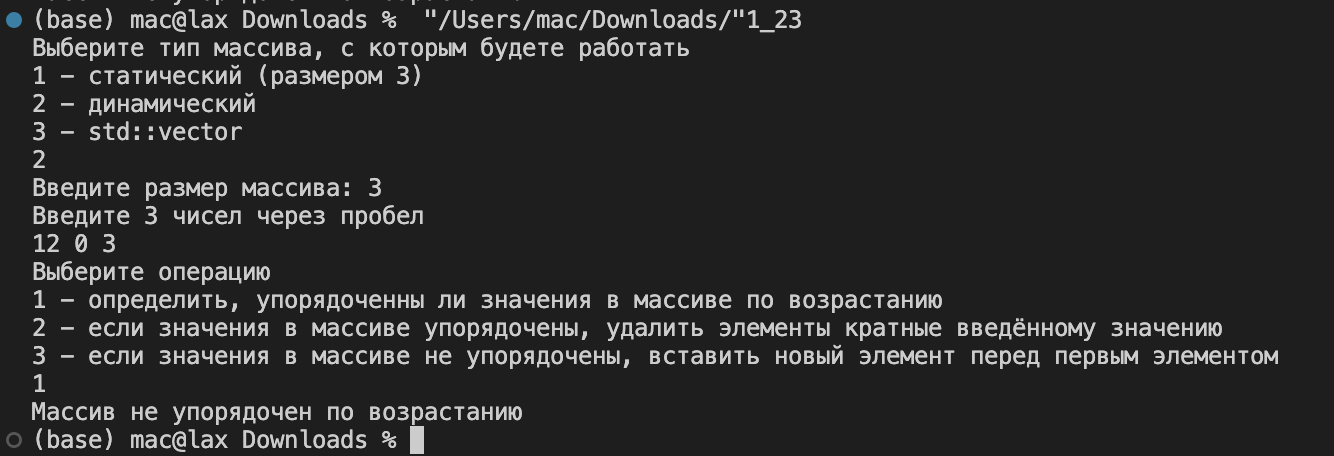


Рисунок 4. Тестирование программы

Протестируем выполнение программой второго упражнения. Для этого поочередно будем выбирать в меню все возможные типы массива, выберем вторую функцию, введем через пробел значения 2 3 4 5 и выберем удалить элементы кратные 2. Результатом должен быть массив 3 5. На рисунках 5, 6, 7 видим, как программа вывела верный ответ.

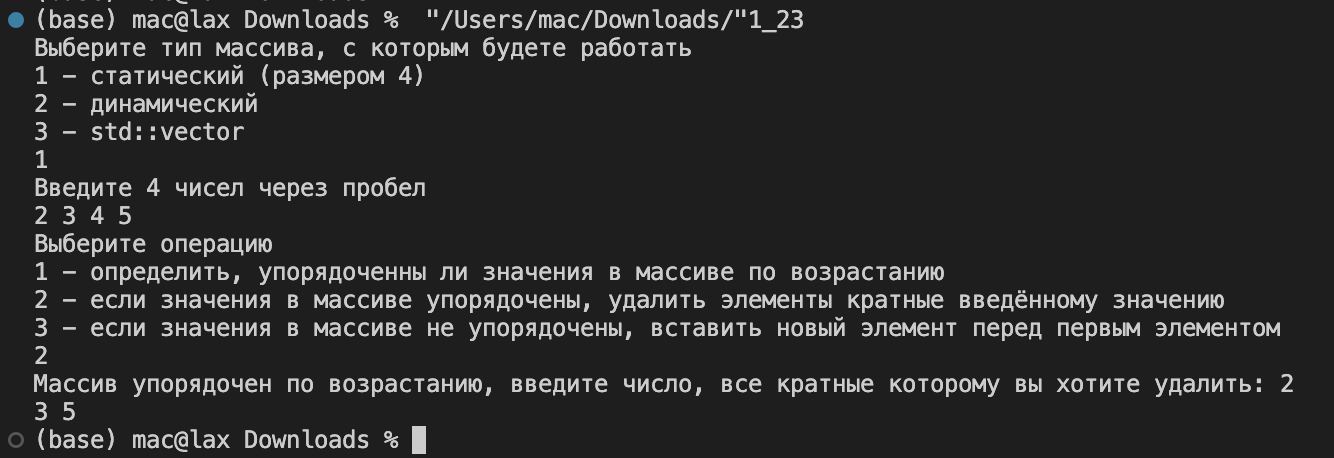
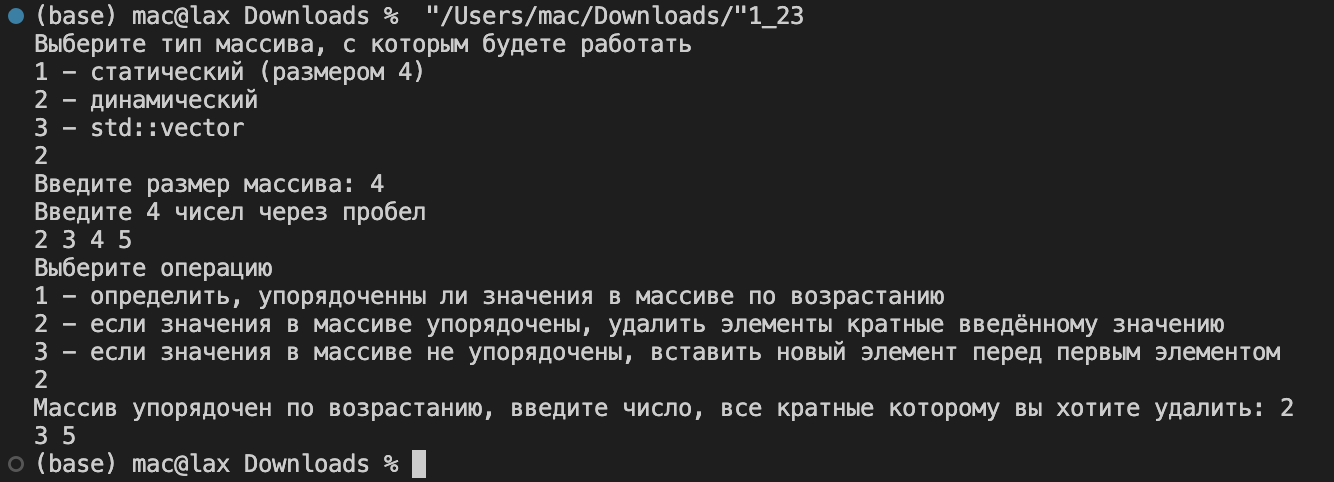


Рисунок 5. Тестирование программы

Рисунок 6. Тестирование программы

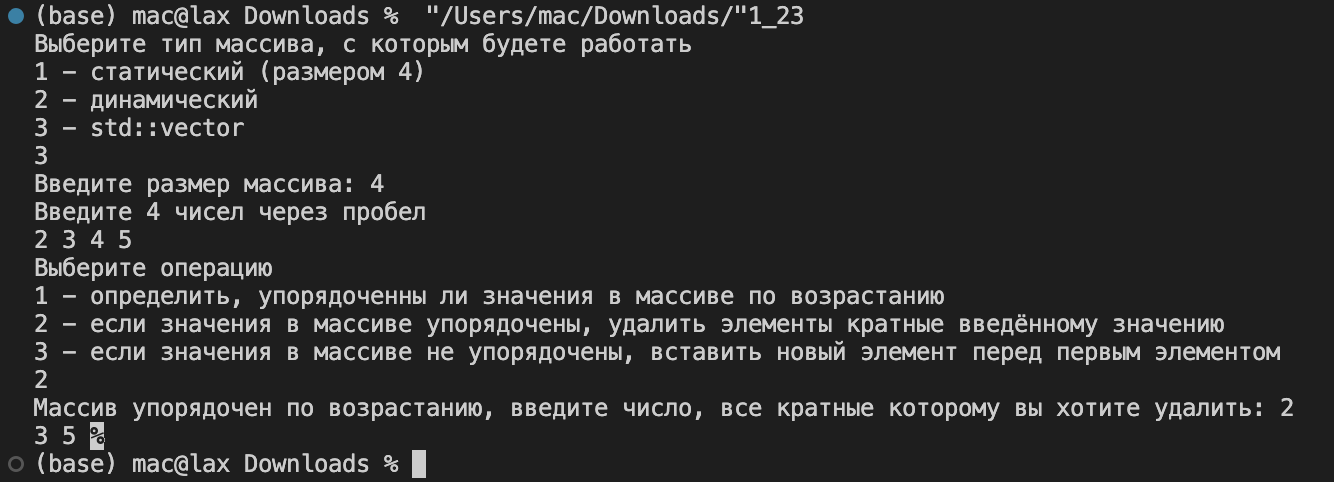


Рисунок 7. Тестирование программы

Протестируем выполнение программой третьего упражнения. Для этого поочередно будем выбирать в меню все возможные типы массива, выберем третью функцию, и введем через пробел значения 4 3 2 1 и выберем вставить в начало число 100. Результатом должен быть массив 100 4 3 2 1. На рисунках 8, 9, 10 видим, как программа вывела верный ответ.

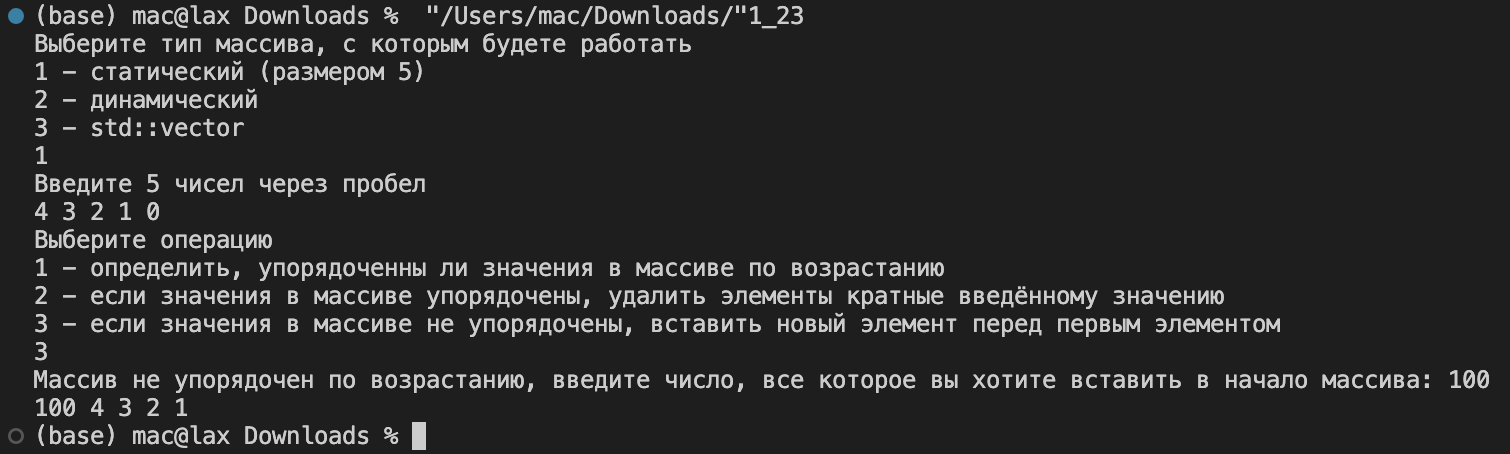
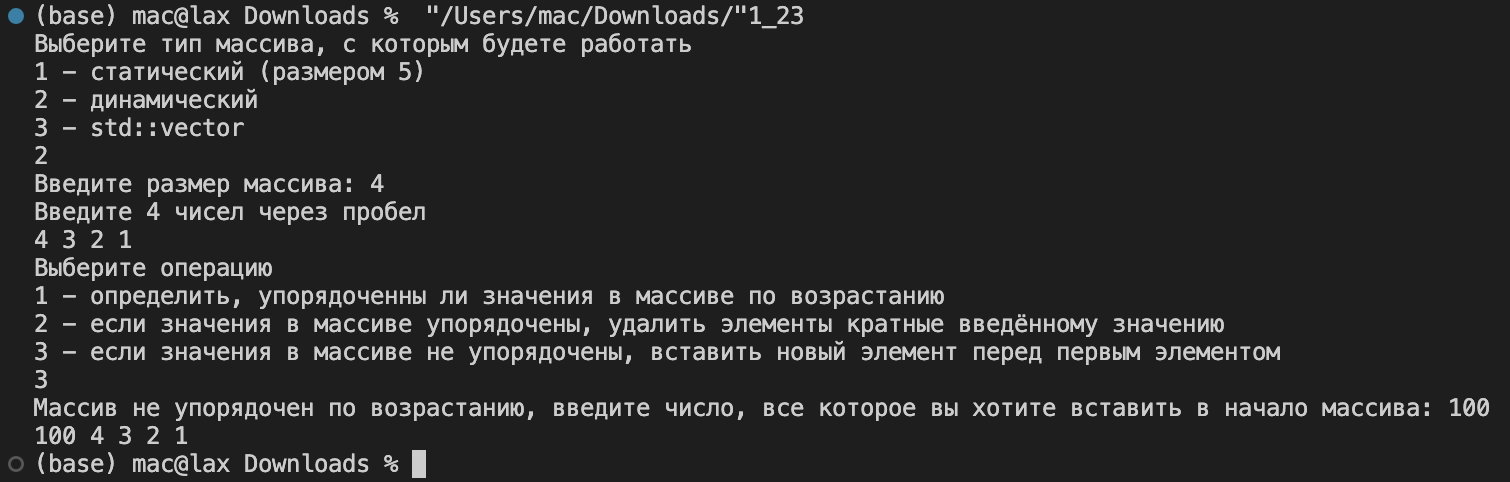


Рисунок 8. Тестирование программы

Рисунок 9. Тестирование программы

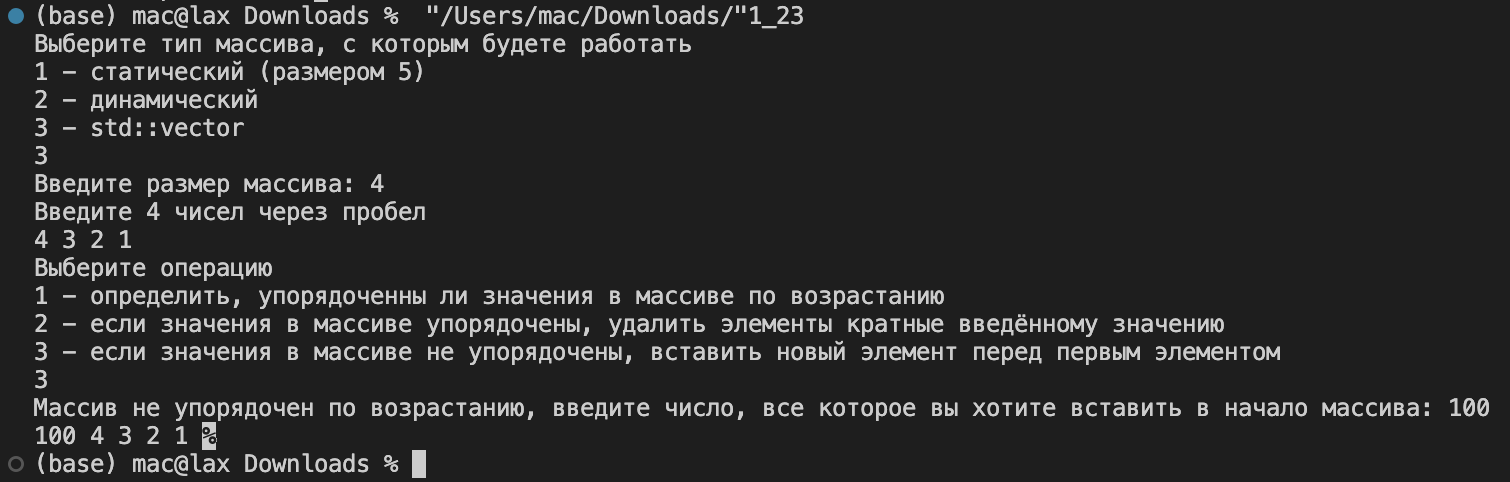


Рисунок 10. Тестирование программы

Из результатов выполнения программы видно:

1. Программа работает корректно, выполняя все поставленные задачи.

# **Вывод**

В результате выполнения работы я:

1. Освоил алгоритмы работы со статическими и динамическими массивами в C++
2. Научился тестированию простых программ

# **Исходный код программы**

|  |
| --- |
| #include <iostream>  #include <vector>  #define STATIC\_ARRAY\_SIZE 10  // Static and dynamic  void read\_array(int array[], int size=STATIC\_ARRAY\_SIZE)  {  for (int i = 0; i < size; i++)  {  std::cin >> array[i];  }  }  void print\_array(int array[], int size=STATIC\_ARRAY\_SIZE)  {  for (int i = 0; i < size; i++)  {  std::cout << array[i] << ' ';  }  std::cout << std::endl;  }  bool is\_sorted(int array[], int size=STATIC\_ARRAY\_SIZE)  {  for (int i = 1; i < size; i++)  {  if (array[i] < array[i - 1])  {  return false;  }  }  return true;  }  // Static  int delete\_multiples(int array[], int num)  {  int deleted = 0;  for (int i = 0; i < STATIC\_ARRAY\_SIZE - deleted; i++)  {  if (array[i] % num == 0)  {  for (int j = i; j < STATIC\_ARRAY\_SIZE - 1; j++)  {  array[j] = array[j + 1];  }  if (STATIC\_ARRAY\_SIZE > 0)  {  array[STATIC\_ARRAY\_SIZE - 1] = 0;  }  deleted++;  i--;  }  }  return deleted;  }  int insert\_first(int array[], int num)  {  for (int i = STATIC\_ARRAY\_SIZE - 1; i > 0; i--)  {  array[i] = array[i - 1];  }  if (STATIC\_ARRAY\_SIZE > 0)  {  array[0] = num;  return 1;  }  return 0;  }  // Dynamic  int delete\_multiples(int\*& array, int size, int num)  {  int deleted = 0;  for (int i = 0; i < size - deleted; i++)  {  if (array[i] % num == 0)  {  for (int j = i; j < size - 1; j++)  {  array[j] = array[j + 1];  }  if (size > 0)  {  array[size - 1] = 0;  }  deleted++;  i--;  }  }  int new\_size = size - deleted;  realloc(array, sizeof(int) \* new\_size);  return deleted;  }  int insert\_first(int\*& array, int size, int num)  {  realloc(array, sizeof(int) \* (size + 1));  for (int i = size; i > 0; i--)  {  array[i] = array[i - 1];  }  array[0] = num;  return 1;  }  // std::vector  void read\_array(std::vector<int>& array)  {  for (int i = 0; i < array.size(); i++)  {  std::cin >> array[i];  }  }  void print\_array(const std::vector<int>& array)  {  for (int i = 0; i < array.size(); i++)  {  std::cout << array[i] << ' ';  }  }  bool is\_sorted(const std::vector<int>& array)  {  for (int i = 1; i < array.size(); i++)  {  if (array[i] < array[i - 1])  {  return false;  }  }  return true;  }  void delete\_multiples(std::vector<int>& array, int num)  {  for (int i = 0; i < array.size(); i++)  {  if (array[i] % num == 0)  {  array.erase(array.begin() + i);  i--;  }  }  }  void insert\_first(std::vector<int>& array, int num)  {  array.push\_back(0);  for (int i = array.size() - 1; i > 0; i--)  {  array[i] = array[i - 1];  }  array[0] = num;  }  int main()  {  int array\_type = 0, operation = 0;  const char\* operation\_prompt =  "Выберите операцию\n"  "1 - определить, упорядоченны ли "  "значения в массиве по возрастанию\n"  "2 - если значения в массиве упорядочены, "  "удалить элементы кратные введённому значению\n"  "3 - если значения в массиве не упорядочены, "  "вставить новый элемент перед первым элементом";    std::cout << "Выберите тип массива, с которым будете работать\n"  << "1 - статический (размером " << STATIC\_ARRAY\_SIZE << ")\n"  << "2 - динамический\n"  << "3 - std::vector\n";  std::cin >> array\_type;  if (array\_type == 1)  {  int static\_array[STATIC\_ARRAY\_SIZE];  std::cout << "Введите " << STATIC\_ARRAY\_SIZE  << " чисел через пробел" << std::endl;  read\_array(static\_array);  std::cout << operation\_prompt << std::endl;  std::cin >> operation;  if (operation == 1)  {  if (is\_sorted(static\_array))  {  std::cout << "Массив упорядочен по возрастанию" << std::endl;  }  else  {  std::cout << "Массив не упорядочен по возрастанию" << std::endl;  }  }  else if (operation == 2)  {  if (is\_sorted(static\_array))  {  int num = 0;  std::cout << "Массив упорядочен по возрастанию, "  << "введите число, все кратные которому вы "  << "хотите удалить: ";  std::cin >> num;  int deleted = delete\_multiples(static\_array, num);  print\_array(static\_array, STATIC\_ARRAY\_SIZE - deleted);  }  else  {  std::cout << "Массив не упорядочен по возрастанию" << std::endl;  print\_array(static\_array, STATIC\_ARRAY\_SIZE);  }  }  else if (operation == 3)  {  if (!is\_sorted(static\_array))  {  int num = 0;  std::cout << "Массив не упорядочен по возрастанию, "  << "введите число, все которое вы хотите "  << "вставить в начало массива: ";  std::cin >> num;  int inserted = insert\_first(static\_array, num);  print\_array(static\_array, STATIC\_ARRAY\_SIZE);  }  else  {  std::cout << "Массив упорядочен по возрастанию" << std::endl;  print\_array(static\_array, STATIC\_ARRAY\_SIZE);  }  }  else  {  std::cout << "Такого варианта нет!";  }  }  else if (array\_type == 2)  {  int size = 0;  std::cout << "Введите размер массива: ";  std::cin >> size;  int\* dynamic\_array = new int[size];  std::cout << "Введите " << size  << " чисел через пробел" << std::endl;  read\_array(dynamic\_array, size);  std::cout << operation\_prompt << std::endl;  std::cin >> operation;  if (operation == 1)  {  if (is\_sorted(dynamic\_array, size))  {  std::cout << "Массив упорядочен по возрастанию" << std::endl;  }  else  {  std::cout << "Массив не упорядочен по возрастанию" << std::endl;  }  }  else if (operation == 2)  {  if (is\_sorted(dynamic\_array, size))  {  int num = 0;  std::cout << "Массив упорядочен по возрастанию, "  << "введите число, все кратные которому вы "  << "хотите удалить: ";  std::cin >> num;  int deleted = delete\_multiples(dynamic\_array, size, num);  print\_array(dynamic\_array, size - deleted);  }  else  {  std::cout << "Массив не упорядочен по возрастанию" << std::endl;  print\_array(dynamic\_array, size);  }  }  else if (operation == 3)  {  if (!is\_sorted(dynamic\_array, size))  {  int num = 0;  std::cout << "Массив не упорядочен по возрастанию, "  << "введите число, все которое вы хотите "  << "вставить в начало массива: ";  std::cin >> num;  int inserted = insert\_first(dynamic\_array, size, num);  print\_array(dynamic\_array, size + inserted);  }  else  {  std::cout << "Массив упорядочен по возрастанию" << std::endl;  print\_array(dynamic\_array, size);  }  }  else  {  std::cout << "Такого варианта нет!";  }  }  else if (array\_type == 3)  {  int size = 0;  std::cout << "Введите размер массива: ";  std::cin >> size;  std::vector<int> vector(size, 0);  std::cout << "Введите " << size  << " чисел через пробел" << std::endl;  read\_array(vector);  std::cout << operation\_prompt << std::endl;  std::cin >> operation;  if (operation == 1)  {  if (is\_sorted(vector))  {  std::cout << "Массив упорядочен по возрастанию" << std::endl;  }  else  {  std::cout << "Массив не упорядочен по возрастанию" << std::endl;  }  }  else if (operation == 2)  {  if (is\_sorted(vector))  {  int num = 0;  std::cout << "Массив упорядочен по возрастанию, "  << "введите число, все кратные которому вы "  << "хотите удалить: ";  std::cin >> num;  delete\_multiples(vector, num);  print\_array(vector);  }  else  {  std::cout << "Массив не упорядочен по возрастанию" << std::endl;  print\_array(vector);  }  }  else if (operation == 3)  {  if (!is\_sorted(vector))  {  int num = 0;  std::cout << "Массив не упорядочен по возрастанию, "  << "введите число, все которое вы хотите "  << "вставить в начало массива: ";  std::cin >> num;  insert\_first(vector, num);  print\_array(vector);  }  else  {  std::cout << "Массив упорядочен по возрастанию" << std::endl;  print\_array(vector);  }  }  else  {  std::cout << "Такого варианта нет!";  }  }  else  {  std::cout << "Такого варианта нет!";  }  return 0;  } |