|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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: «Одномерный массив»** | |
|  | |
| Выполнил студент группы ИКБО-07-22 | Козлов К.И. |
| Принял преподаватель | Филатов А.С. |

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

Москва 2023

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

Требования к упражнениям:

Вариант №26. Условие задания:

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

5. Провести тестирование программы на небольших объемах данных, введенных вручную.

6. Составить отчет, отобразив в нем описание выполнения всех этапов разработки, тестирования и код всей программы со скриншотами результатов тестирования.

# **Решение**

В данной практической работе была проведена работа с разными видами массивов: статическим массивом, динамическим массивом и вектором.

Массив — упорядоченная последовательность данных, состоящая из конечного числа элементов, имеющих один и тот же тип, и обозначаемая одним именем. 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.

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. Возможность изменения размера отличает динамический массив от статического, размер которого задаётся на момент компиляции программы. Динамические массивы дают возможность более гибкой работы с данными, так как позволяют не прогнозировать хранимые объёмы данных, а регулировать размер массива в соответствии с реально необходимыми объёмами.

Динамический массив в языке C++ определяется через указатель.

Указатель - переменная, в которой хранится адрес памяти объекта. Указатели широко используются в C++ для трех основных целей:

- чтобы выделить новые объекты в куче (куча - это хранилище памяти, также расположенное в ОЗУ, которое допускает динамическое выделение памяти и не работает по принципу стека);

- для передачи функций другим функциям;

- для итерации элементов в массивах или других структурах данных.

Контейнер <vector> - современная реализация динамических массивов.

Векторы - это реализация динамического массива в C++, то есть такого массива, чья длина меняется в ходе программы. Называется вектор, потому что добавлять и удалять элементы можно только с одного конца. Шаблон vector расположен в заголовочном файле <vector>. Как и все стандартные компоненты, он расположен в пространстве имён std. Все элементы вектора должны принадлежать одному типу. Например, нельзя совместно хранить данные типов char и int в одном экземпляре вектора.

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.

Поскольку адреса элементов в течение этого процесса меняются, любые ссылки или итераторы элементов в векторе могут стать недействительными. Использование недействительных ссылок приводит к неопределённому поведению.

Для решения первого упражнения было написано несколько функций: Choice(), Increasing() (для статического массива); Dy\_Choice(), Dy\_Increasing() (для динамического массива); Vec\_Choice(), Vector\_Increasing() (для вектора). Данные функции определяют, упорядочены ли значения в массиве по возрастанию.

Функция Choice() имеет по пять входных параметров: int Quantity, int Array[], int Counter, int Another\_Counter, int Ch. Некоторые из них используются в роли счётчиков, чтобы по итогу определить, как расположены элементы в статическом массиве: возрастают, убывают и т.д.

|  |
| --- |
| void Choice(int Quantity, int Array[], int Counter, int Another\_Counter, int Ch)  {  while (true)  {  int choice = 0;  cout << "Что вы хотите сделать с массивом?\n\nВведите \"1\", чтобы проверить элементы массива на ВОЗРАСТАНИЕ;\nВведите \"2\", чтобы удалить элементы из массива " <<  "(работает только если значения в массиве НЕ УПОРЯДОЧЕНЫ ПО ВОЗРАСТАНИЮ.\nВ этом случае вам будет предложено ввести новое числовое значение, чтобы из массива удалились " <<  "элементы кратные этому значению);\nВведите \"3\", чтобы вставить новое числовое значение перед самым первым элементом большим его по значению (работает только если значения в" <<  "массиве УПОРЯДОЧЕНЫ ПО ВОЗРАСТАНИЮ);\nВведите \"4\", чтобы выйти из программы.\n\nВы хотите: ";  cin >> choice;  while ((choice < 1) || (choice > 4))  {  cout << "Вы ввели некорректное значение! Повторите попытку ввода: ";  cin >> choice;  }  cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";  switch (choice)  {  case 1: // Проверка на возрастание  {  Ch = 1;  cout << "\nВы хотите проверить элементы массива на ВОЗРАСТАНИЕ.";  cout << endl << "\nВаш массив значений имеет вид: "; // Вывод массива значений  for (int i = 0; i < Quantity; i++)  {  if (Array[i] != -858993460)  {  cout << Array[i] << " ";  }  }  Increasing(Quantity, Array, Counter, Another\_Counter, Ch);  cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n";  break;  }  case 2: // Удалить элемент из массива  {  Ch = 2;  cout << "\nВы хотите УДАЛИТЬ элементы из массива.\nСначала проверим НЕ УПОРЯДОЧЕНЫ ли значения в массиве по возрастанию?";  cout << endl << "\nВаш массив значений имеет вид: "; // Вывод массива значений  for (int i = 0; i < Quantity; i++)  {  if (Array[i] != -858993460)  {  cout << Array[i] << " ";  }  }  Increasing(Quantity, Array, Counter, Another\_Counter, Ch);  break;  }  case 3: // Добавить элемент в массив  {  Ch = 3;  cout << "\nВы хотите ДОБАВИТЬ элемент в массив.\nСначала проверим УПОРЯДОЧЕНЫ ли значения в массиве по возрастанию?";  cout << endl << "\nВаш массив значений имеет вид: "; // Вывод массива значений  for (int i = 0; i < Quantity; i++)  {  if (Array[i] != -858993460)  {  cout << Array[i] << " ";  }  }  Increasing(Quantity, Array, Counter, Another\_Counter, Ch);  break;  }  case 4:  {  cout << "Вы выбрали выход из программы!";  cout << "\n\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n";  goto Exit;  }  break;  }  continue;  }  Exit:  {  ;  }  } |

Описание функции

|  |
| --- |
| Код функции |

Описание функции

|  |
| --- |
| Код функции |

Описание функции

|  |
| --- |
| Код функции |

Описание функции

|  |
| --- |
| Код функции |

Описание функции

|  |
| --- |
| Код функции |

Описание функции

|  |
| --- |
| Код функции |

Описание функции

|  |
| --- |
| Код функции |

Описание функции

|  |
| --- |
| Код функции |

Описание функции

|  |
| --- |
| Код функции |

Описание функции

|  |
| --- |
| Код функции |

Описание функции

|  |
| --- |
| Код функции |

При запуске программы пользователь видит предупреждение, пользовательское меню, позволяющее выбирать с каким массивом продолжить работу.

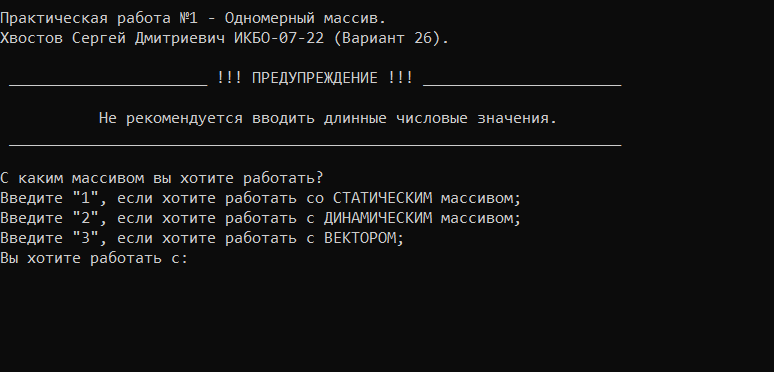


Рисунок 1. Интерфейс программы

# **4. Тестирование**

Протестируем работу программы. Для этого выберем в меню 1 – работу со статическим массивом. Вводим размер массива 6. Заполняем массив. Затем пользователю будет дан выбор, что делать с массивом дальше: проверить на возрастание, удалить элемент из массива, добавить новое значение в массив или выйти из программы. Вводим 1, чтобы проверить массив на возрастание. Программа должна сообщить нам, что элементы в массиве расположены в порядке возрастания. После успешного выполнения, пользователю повторно будет предложено, что делать с массивом. На рисунке 2 видно, как программа вывела корректный результат.

Рисунок 2. Проверка статического массива на возрастание

Описание теста.

Рисунок 3. Добавление нового элемента в динамический массив (часть 1)

Рисунок 4. Добавление нового элемента в динамический массив (часть 2)

Описание теста.

Рисунок 5. Удаление элементов из вектора

# **5. Выводы**

В результате выполнения работы я:

1. Освоил алгоритмы работы со статическими, с динамическими массивами, с векторами.
2. Научился проводить операции над векторами и разными видами массивов.
3. Реализовал работу с массивами и векторами на языке программирования C++.
4. Приобрел навыки по работе с массивами и векторами.

# **6. Исходный код программы**

|  |
| --- |
| Код программы |