|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | 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 №9. Условие задания:

|  |  |
| --- | --- |
| Упражнение 1 | Найти индекс элемента массива, которое является палиндромом. |
| Упражнение 2 | Удалить элементы массива, расположенное непосредственно перед элементом, содержащим число палиндром. |
| Упражнение 3 | Вставить новый элемент в массив после элемента массива, который является палиндромом. |

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**

Массив - структура данных, хранящая набор значений (элементов массива), идентифицируемых по индексу

Статический массив – это структура данных, которая создается во время компиляции. Размер массива нельзя изменять во время работы программы.

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

Динамическим называется массив, размер которого может изменяться во время исполнения программы. Динамические массивы дают возможность более гибкой работы с данными.

Динамический массив в языке C++ определяется через указатель.

Указатель - переменная, в которой хранится адрес памяти объекта.

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.

Для решения первого упражнения было написано несколько функций: Palindrome () (Для всех видов массивов). ShowPalindromes(), EnterDynamic() (для статического и динамического массивов, причем для всех упражнений). ShowPalindromeVector() (для вектора). Данные функции позволяют найти индексы элементов, которые являются палиндромами.

Функция Palindrome() имеет один входной параметр - string x. Это строчное представление числа, с помощью которого функция определяет, является ли число палиндромом.

|  |
| --- |
| bool Palindrome(string x)  {  if (x.length() >= 2)  {  if (x[0] == '-')  {  x = x + '-';  }  for (int i = 0; i < x.length() / 2; i++)  {  if (x[i] != x[x.length() - i - 1])  {  return (false);  }  }  return (true);  }  else  {  return (false);  }  } |

Функция EnterDynamic() имеет два параметра: это указатель a и целочисленная переменная l. Функция выполняет действие заполнения массива пользователем вручную.

|  |
| --- |
| void EnterDynamic(int\* a, int L)  {  for (int i = 0; i < L; i++)  {  cin >> a[i];  }  } |

Функция ShowPalindromes() также имеет два параметра: это указатель a и целочисленная переменная l. Функция выводит на экран номера элементов, которые являются палиндромами, если такие введены пользователем.

|  |
| --- |
| void ShowPalindromes(int\* a, int l)  {  cout << "There are some Palindromes indexes...";  for (int i = 0; i < l; i++)  {  if (Palindrome(to\_string(a[i])))  {  cout << i << ' ';  }  }  cout << endl;  } |

ShowPalindromesVector() идентична предыдущей функции, за исключением того, что на вход вместо указателя получает вектор a вместе с целочисленной переменной l.

|  |
| --- |
| void ShowPalindromeVector(vector <int> a, int l)  {  cout << "There are some Palindromes indexes...";  for (int i = 0; i < l; i++)  {  if (Palindrome(to\_string(a[i])))  {  cout << i << ' ';  }  }  cout << endl;  } |

Для решения второго упражнения были написаны следующие функции: FindPalindrome(), ArrayDeletePrePalindrome()(для статического и динамического массивов), а так же FindPalindromeVector и ArrayDeletePrePalindromeVector() (для вектора).

FindPalindrome() имеет два входных параметра: указатель a и целочисленную переменную l, а также возвращает целочисленное значение – индекс элемента палиндрома, если такой не нашелся, возвращает -1.

|  |
| --- |
| int FindPalindrome(int\* a, int l)  {  for (int i = 0; i < l; i++)  {  if (Palindrome(to\_string(a[i])))  {  return (i);  }  }  return (-1);  } |

FindPalindromeVector() имеет два входных параметра: вектор a и целочисленную переменную l, а также возвращает целочисленное значение – индекс элемента палиндрома, если такой не нашелся, возвращает -1.

|  |
| --- |
| int FindPalindromesVector(vector <int> a, int l)  {  for (int i = 0; i < l; i++)  {  if (Palindrome(to\_string(a[i])))  {  return (i);  }  }  return (-1);  } |

ArrayDeletePrePalindrome() имеет три водных параметра: указатель a, целочисленную переменную l и целочисленную переменную Index. Функция сдвигает элементы перед палиндромом и не сохраняет их значения.

|  |
| --- |
| void ArrayDeletePrePalindrome(int\* a, int l, int Index)  {  for (int i = Index; i < l; i++)  {  a[i - Index] = a[i];  }  } |

ArrayDeletePrePalindromeVector() имеет три водных параметра: вектор a, целочисленную переменную l и целочисленную переменную Index. Функция возвращает новый вектор без элементов перед найденным элементом палиндромом, а также реализует сдвиг этих элементов перед палиндромом и не сохраняет их значения.

|  |
| --- |
| vector <int> ArrayDeletePrePalindromeVector(vector <int> a, int l, int Index)  {  for (int i = Index; i < l; i++)  {  a[i - Index] = a[i];  }  return a;  } |

Для решения третьего упражнения были написаны следующие функции: ArrayAppendPostPalindrome() (для статического и динамического массивов) и ArrayAppendPostPalindromeVector() (для вектора).

ArrayAppendPostPalindrome() имеет три вохдных параметра: имеет три водных параметра: указатель a, целочисленную переменную l и Index. Эта функция сдвигает все элементы начиная с предпоследнего на одну позицию вперед и позволяет пользователю вписать в освободившееся место новый элемент.

|  |
| --- |
| void ArrayAppendPostPalindrome(int\* a, int l, int Index)  {  int App;  cin >> App;  for (int i = l + 1; i > Index; i--)  {  a[i] = a[i - 1];  }  a[Index + 1] = App;  } |

ArrayAppendPostPalindromeVector() имеет три водных параметра: вектор a, целочисленную переменную l и целочисленную переменную Index. Данная функция возвращает новый вектор, в котором на один элемент больше. Этот элемент ввел пользователь в освободившееся место после сдвига, который также реализует эта функция (по аналогии с предыдущей).

|  |
| --- |
| vector <int> ArrayAppendPostPalindromeVector(vector <int> a, int l, int Index)  {  int App;  cin >> App;  for (int i = l - 1; i > Index; i--)  {  a[i] = a[i - 1];  }  a[Index + 1] = App;  return a;  } |

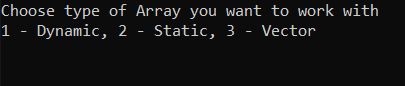
При запуске программы пользователь видит сообщение, о том, что он может выбрать тип массива или вектор, с которым хочет работать.

Рисунок 1. Интерфейс программы

# **4. Тестирование**

Тестирование работы программы. Для этого введем 1 – режим работы с динамическим массивом. Далее вводим длину массива (7) и заполняем его. После этого есть выбор операций над массивом: удаление элементов перед палиндромом, добавление элемента после палиндрома, вывод индексов палиндромов. Выберем 1 – Показать индексы элементов палиндромов.

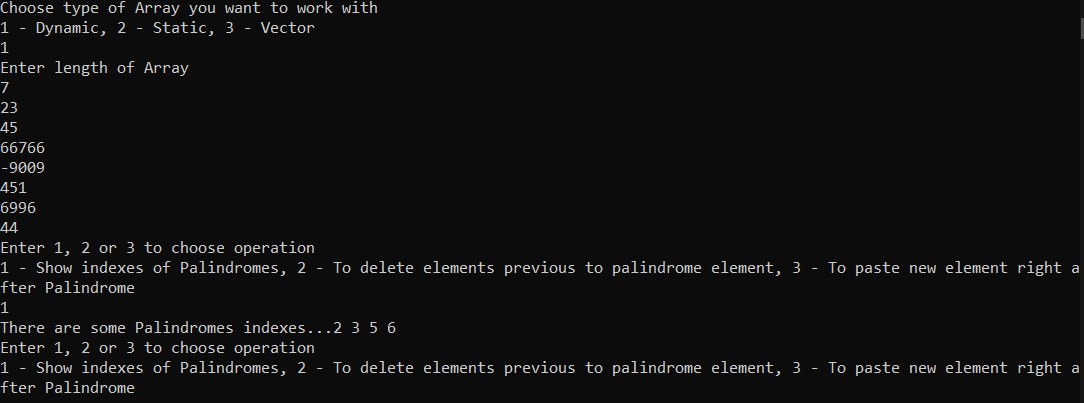


Рисунок 2. Нахождение индексов элементов палиндромов в динамическом массиве

Проверка выполнение второго упражнения уже на примере вектора. Введем вектор с длинной 6 и выберем вторую операцию: удаление элементов перед элементом палиндромом.

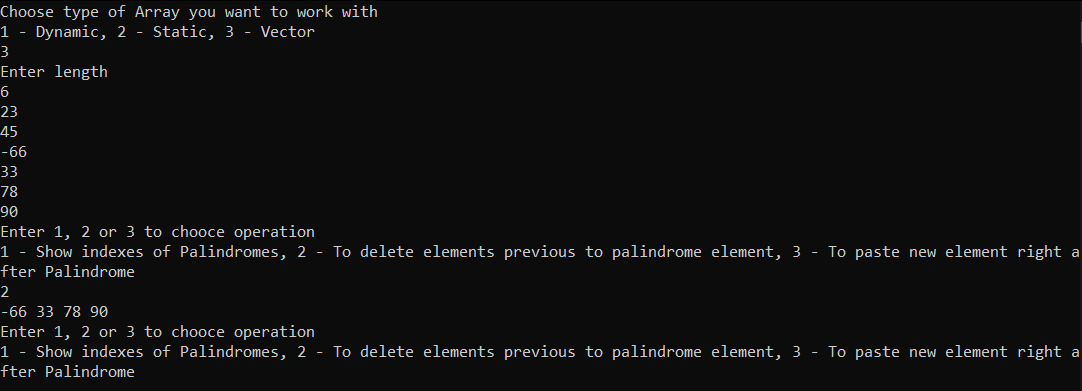


Рисунок 3. Удаление элементов вектора, стоящих перед элементом палиндромом.

Проверка решения третьего упражнения на примере статического массива. Введем статический массив с длинной 10 и вставим элемент со значением 451 после первого встреченного палиндрома.

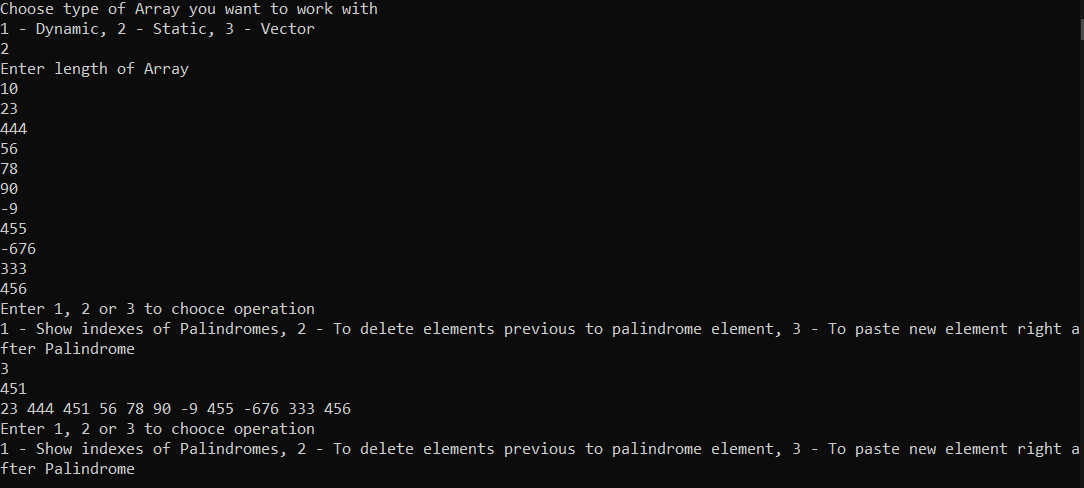


Рисунок 5. Удаление элементов из вектора

# **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**

|  |
| --- |
| #include <iostream>  #include <vector>  #include <string>  #include <cmath>  using namespace std;  void EnterDynamic(int\* a, int L)  {  for (int i = 0; i < L; i++)  {  cin >> a[i];  }  }  void ShowDynamic(int\* a, int L)  {  for (int i = 0; i < L; i++)  {  cout << a[i] << ' ';  }  cout << endl;  }  void ShowVector(vector <int> a, int l)  {  for (int i = 0; i < l; i++)  {  cout << a[i] << ' ';  }  cout << endl;  }  void EnterVector(vector <int> a, int L)  {  for (int i = 0; i < L; i++)  {  cin >> a[i];  }  }  bool Palindrome(string x)  {  if (x.length() >= 2)  {  if (x[0] == '-')  {  x = x + '-';  }  for (int i = 0; i < x.length() / 2; i++)  {  if (x[i] != x[x.length() - i - 1])  {  return (false);  }  }  return (true);  }  else  {  return (false);  }  }  void ShowPalindromes(int\* a, int l)  {  cout << "There are some Palindromes indexes...";  for (int i = 0; i < l; i++)  {  if (Palindrome(to\_string(a[i])))  {  cout << i << ' ';  }  }  cout << endl;  }  int FindPalindrome(int\* a, int l)  {  for (int i = 0; i < l; i++)  {  if (Palindrome(to\_string(a[i])))  {  return (i);  }  }  return (-1);  }  int FindPalindromesVector(vector <int> a, int l)  {  for (int i = 0; i < l; i++)  {  if (Palindrome(to\_string(a[i])))  {  return (i);  }  }  return (-1);  }  void ShowPalindromeVector(vector <int> a, int l)  {  cout << "There are some Palindromes indexes...";  for (int i = 0; i < l; i++)  {  if (Palindrome(to\_string(a[i])))  {  cout << i << ' ';  }  }  cout << endl;  }  void ArrayDeletePrePalindrome(int\* a, int l, int Index)  {  for (int i = Index; i < l; i++)  {  a[i - Index] = a[i];  }  }  vector <int> ArrayDeletePrePalindromeVector(vector <int> a, int l, int Index)  {  for (int i = Index; i < l; i++)  {  a[i - Index] = a[i];  }  return a;  }  void ArrayAppendPostPalindrome(int\* a, int l, int Index)  {  int App;  cin >> App;  for (int i = l + 1; i > Index; i--)  {  a[i] = a[i - 1];  }  a[Index + 1] = App;  }  vector <int> ArrayAppendPostPalindromeVector(vector <int> a, int l, int Index)  {  int App;  cin >> App;  for (int i = l - 1; i > Index; i--)  {  a[i] = a[i - 1];  }  a[Index + 1] = App;  return a;  }  int main()  {  int ChoiceArray;  cout << "Choose type of Array you want to work with" << endl;  cout << "1 - Dynamic, 2 - Static, 3 - Vector" << endl;  cin >> ChoiceArray;  if (ChoiceArray == 1)  {  int ChoiceOperation;  int Length;  cout << "Enter length of Array" << endl;  cin >> Length;  int\* a = (int\*)malloc(sizeof(int) \* Length);  EnterDynamic(a, Length);  do  {  cout << "Enter 1, 2 or 3 to choose operation" << endl;  cout << "1 - Show indexes of Palindromes, 2 - To delete elements previous to palindrome element, 3 - To paste new element right after Palindrome" << endl;  cin >> ChoiceOperation;  switch (ChoiceOperation)  {  case 1:  {  ShowPalindromes(a, Length);  }  break;  case 2:  {  int PalindromeIndex = FindPalindrome(a, Length);  if (PalindromeIndex != -1)  {  ArrayDeletePrePalindrome(a, Length, PalindromeIndex);  Length -= PalindromeIndex;  ShowDynamic(a, Length);  a = (int\*)realloc(a, sizeof(int) \* Length);  }  break;  }  case 3:  {  int PalindromeIndex = FindPalindrome(a, Length);  if (PalindromeIndex != -1)  {  Length += 1;  a = (int\*)realloc(a, sizeof(int) \* Length);  ArrayAppendPostPalindrome(a, Length, PalindromeIndex);  ShowDynamic(a, Length);  }  }  break;  }  } while (ChoiceOperation == 1 || ChoiceOperation == 2 || ChoiceOperation == 3);  }  else if (ChoiceArray == 2)  {  int ChoiceOperation;  int Length;  cout << "Enter length of Array" << endl;  cin >> Length;  int a[100];  EnterDynamic(a, Length);  do  {  cout << "Enter 1, 2 or 3 to chooce operation" << endl;  cout << "1 - Show indexes of Palindromes, 2 - To delete elements previous to palindrome element, 3 - To paste new element right after Palindrome" << endl;  cin >> ChoiceOperation;  switch (ChoiceOperation)  {  case 1:  {  ShowPalindromes(a, Length);  }  break;  case 2:  {  int PalindromeIndex = FindPalindrome(a, Length);  if (PalindromeIndex != -1)  {  ArrayDeletePrePalindrome(a, Length, PalindromeIndex);  Length -= PalindromeIndex;  ShowDynamic(a, Length);  }  }  break;  case 3:  {  int PalindromeIndex = FindPalindrome(a, Length);  if (PalindromeIndex != -1)  {  Length += 1;  ArrayAppendPostPalindrome(a, Length, PalindromeIndex);  ShowDynamic(a, Length);  }  }  break;  }  } while (ChoiceOperation == 1 || ChoiceOperation == 2 || ChoiceOperation == 3);  }  else if (ChoiceArray == 3)  {  int ChoiceOperation;  int Length;  cout << "Enter length" << endl;  cin >> Length;  vector <int> a(Length);  for (int i = 0; i < Length; i++)  {  cin >> a[i];  }  do  {  cout << "Enter 1, 2 or 3 to chooce operation" << endl;  cout << "1 - Show indexes of Palindromes, 2 - To delete elements previous to palindrome element, 3 - To paste new element right after Palindrome" << endl;  cin >> ChoiceOperation;  switch (ChoiceOperation)  {  case 1:  {  ShowPalindromeVector(a, Length);  }  break;  case 2:  {  int PalindromeIndex = FindPalindromesVector(a, Length);  if (PalindromeIndex != -1)  {  a = ArrayDeletePrePalindromeVector(a, Length, PalindromeIndex);  Length -= PalindromeIndex;  a.resize(Length);  }  ShowVector(a, Length);  }  break;  case 3:  {  int PalindromeIndex = FindPalindromesVector(a, Length);  if (PalindromeIndex != -1)  {  Length += 1;  a.resize(Length);  a = ArrayAppendPostPalindromeVector(a, Length, PalindromeIndex);  ShowVector(a, Length);  }  }  break;  }  } while (ChoiceOperation == 1 || ChoiceOperation == 2 || ChoiceOperation == 3);  }  } |