

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

# ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 2

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Абстрактный тип данных и его реализация на

одномерном статическом массиве

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-65-23 |  | Петров А. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

# **Оглавление**

[1. УСЛОВИЕ ЗАДАЧИ. 2](#_Toc161869645)

[2. РАЗРАБОТКА РЕШЕНИЯ 5](#_Toc161869646)

[2.1. Разработка абстрактного типа данных задачи, используя шаблон 5](#_Toc161869647)

[2.2. Разработка и запись на псевдокоде алгоритмов операций 7](#_Toc161869648)

[2.3. Код алгоритма на языке С++. 11](#_Toc161869649)

[2.4. Тестирование алгоритма. 14](#_Toc161869650)

[3. Выводы 23](#_Toc161869651)

[4. Информационные источники 24](#_Toc161869652)

# УСЛОВИЕ ЗАДАЧИ.

**Задание**

Дано множество из n целых чисел. Дан набор задач (операций), которые

требуется выполнить над исходным множеством. Набор задач определен в

варианте задания табл. 6.

Разработать и реализовать АТД задачи, по управлению множеством посредством операций, указанных в варианте задания. В АТД включить операции по заполнению исходного множества и отображения множества. При разработке алгоритмов операций варианта могут быть выявлены дополнительные алгоритмы, например такие: определить является ли число простым, или определить сумму цифр числа, эти алгоритмы надо включить в раздел операций АТД.

**Требования к выполнению практической работы**

1. Разработать абстрактный тип данных задачи вашего варианта, используя

шаблон:

АТД Имя\_АТД

{

Данные (описание свойств структуры данных задачи).

n – количество элементов множества.

А – список значений элементов множества.

pos – позиция элемента.

*Примечание*. Имена данных могут быть другими.

Операции (объявления операций).

Общие для всех вариантов.

1) заполнение структуры данных значениями (с клавиатуры,

применение датчика случайных чисел);

2) вывод структуры в консоль;

3) удалить элемент в заданной позиции;

4) вставить элемент в заданную позицию.

Дополнительные операции.

Операции, определенные в варианте и выявленные разработчиком

как вспомогательные, для улучшения структуризации кода операции.

}

Реализовать список А из АТД можно на трех видах структур данных: статический массив, динамический массив, vector. В задании 1 рассмотрим реализацию на статическом массиве.

2. Реализовать АТД задачи, используя для представления списка значений

множества А статический массив максимального размера N.

2.1. Выполнить реализацию множества, используя тип данных struct языка С++, включив поля: статический массив для хранения значений множества, поле (например, с именем n) для хранения текущего размера множества, поле (например, с именем N) для хранения размера статического массива. При этом n≤N.

2.2. Разработать и записать на псевдокоде алгоритмы операций:

1) удалить элемент в заданной позиции (pos<=n);

2) вставить элемент в заданную позицию (pos<=n).

2.3. Реализовать все операции, заявленные в АТД, на реализованной структуре представления данных. Каждая операция оформляется отдельной функцией с параметрами.

*Примечание*. Реализацию АТД можно выполнить в отдельном файле заголовка, разрабатываемого проекта. В основной программе его нужно будет подключить. 3. Разработать наборы тестов для тестирования дополнительных операций. Набор тестов для каждой операции оформить отдельной таблицей по формату.

4. Разработать основную программу, демонстрирующую выполнение всех операций над структурой данных на подготовленных тестах. Программа должна управлять всем процессом посредством текстового меню.

5. Выполнить тестирование основной программы на подготовленных тестах.

6. Оформить отчет.

**Вариант задачи № 22**

1. Найти позицию элемента массива значение которого делится на каждую из цифр числа.\*

2. Вставить в массив новый элемент после элемента, значение которого делится на каждую цифру значения.

3. Удалить из массива все элементы, кратные трем.

# РАЗРАБОТКА РЕШЕНИЯ

## 2.1. Разработка абстрактного типа данных задачи, используя шаблон

|  |
| --- |
| АТД MyStruct  {  Данные:  n – количество элементов множества.  А – список значений элементов множества.  typeA – тип множества.  typeitem – тип элемента множества A. |

Операции:

|  |
| --- |
| **Операция 1**  //Операция: заполнение структуры данных значениями.  //Предусловие: 0||1 – входные данные.  //Постусловие: результат при успешном заполнении – 1, при безуспешном код завершения -1.  create( (0||1 – выбор заполнения с клавиатуры или с применение датчика случайных чисел));.- заголовок |

|  |
| --- |
| **Операция 2**  //Операция: вывод структуры в консоль.  //Предусловие: n>0.  //Постусловие: вывод множества A  print(A);.- заголовок |

|  |
| --- |
| **Операция 3**  //Операция: удалить элемент в заданной позиции.  //Предусловие: pos – входные данные, n>0.  //Постусловие: результат при успешном поиске – 0, при безуспешном код завершения -1.  deletePos(A, pos);.- заголовок |

|  |
| --- |
| **Операция 4**  //Операция: вставить элемент в заданную позицию.  //Предусловие: pos, element – входные данные,n>0.  //Постусловие: результат при успешном поиске – element, при безуспешном код завершения -1.  addPos(A, pos, element);.- заголовок |

|  |
| --- |
| **Операция 5**  //Операция: Найти позицию элемента массива значение которого делится на каждую из цифр числа.\*  //Предусловие: temp2 – входные данные,n>0.  //Постусловие: результат при успешном поиске – pos, при безуспешном код завершения -1.  findElement(A, temp2);.- заголовок |

|  |
| --- |
| **Операция 6**  //Операция: Вставить в массив новый элемент после элемента, значение которого делится на каждую цифру значения.  //Предусловие: element, temp2 – входные данные,n>0.  //Постусловие: результат при успешном поиске – (pos+1), при безуспешном код завершения -1.  addElement (A, temp2, element,);.- заголовок |

|  |
| --- |
| **Операция 7**  //Операция: Удалить из массива все элементы, кратные трем.  //Предусловие: n>0.  //Постусловие: результат при успешном удаление – 1, при безуспешном код завершения -1.  deleteKrat3(A);.- заголовок |

## 2.2. Разработка и запись на псевдокоде алгоритмов операций

Таблица 1. Форма представления операции 1 при получении функции зависимости количества выполняемых инструкций от размера задачи

|  |  |  |
| --- | --- | --- |
| Номер инструкции | Код алгоритма create | Количество операций в инструкции |
|  | temp←0 | 1 |
|  | N←20 | 1 |
|  | If (temp == 0) | 1 |
|  | For i←0 to n do | 1+3(n+1) |
|  | temp←cin | n |
|  | A[i]←temp1 | 2n |
|  | od | - |
|  | For i←n to N do | 1+3(n+1) |
|  | A[i]←0 | 2n |
|  | od | - |
|  | else if (temp == 1) | 1 |
|  | For i←0 to n do | 1+3(n+1) |
|  | A[i] = rand(); | 3n |
|  | od | - |
|  | else | - |
|  | return -1 | 1 |

T(n) = 17+17n

Асимптотическая временная сложность линейная – O(n).

Таблица 2. Форма представления операции 2 при получении функции зависимости количества выполняемых инструкций от размера задачи

|  |  |  |
| --- | --- | --- |
| Номер инструкции | Код алгоритма print | Количество операций в инструкции |
|  | pos←0 | 1 |
|  | For i←0 to n do | 1+3(n+1) |
|  | cout << "Элемент " << pos << " = " << A[pos] << endl; | 6n |
|  | od | - |

T(n) = 5+9n

Асимптотическая временная сложность линейная – O(n).

Таблица 3. Форма представления операции 3 при получении функции зависимости количества выполняемых инструкций от размера задачи

|  |  |  |
| --- | --- | --- |
| Номер инструкции | Код алгоритма deletePos | Количество операций в инструкции |
|  | pos←cin | 2 |
|  | for i←pos to n do | 1+3(n-pos+1) |
|  | A[i] = A[i+1] | 3(n-pos) |
|  | od | - |
|  | n--; | 1 |
|  | return 0 | 1 |

T(n) = 6+6n-6pos

Асимптотическая временная сложности – O(n-pos).

Таблица 4. Форма представления операции 4 при получении функции зависимости количества выполняемых инструкций от размера задачи

|  |  |  |
| --- | --- | --- |
| Номер инструкции | Код алгоритма addPos | Количество операций в инструкции |
|  | n++; | 1 |
|  | pos←cin | 2 |
|  | element←cin | 2 |
|  | temp1← A[pos] | 2 |
|  | A[pos]← element | 2 |
|  | for i←pos+1 to n do | 1+1+3(n-pos+1) |
|  | temp2 ← element | (n-pos) |
|  | A[i] ← temp1 | 2(n-pos) |
|  | temp1 ← A[i+1] | 3(n-pos) |
|  | temp1 ← temp2 | (n-pos) |
|  | od |  |
|  | return 0 | 1 |

T(n) = 14+10n-10pos

Асимптотическая временная сложности - O(n-pos).

Таблица 5. Форма представления операции 5 при получении функции зависимости количества выполняемых инструкций от размера задачи

|  |  |  |
| --- | --- | --- |
| Номер инструкции | Код алгоритма findElement | Количество операций в инструкции |
|  | temp2←cin | 2 |
|  | For pos←0 to n do | 1+3(n+1) |
|  | For i←10; temp !=0; i \*= 10 do | n(1+3(n+1)) |
|  | temp1 ← temp2 % i |  |
|  | if ((A[pos] % temp1) != 0) |  |
|  | break; |  |
|  | if ((A[pos] % temp1 == 0) && (temp1 < 10)) |  |
|  | return pos |  |
|  | endif | - |

T(n) = 6+5n+14

Асимптотическая временная сложность квадратичная – O().

Таблица 6. Форма представления операции 6 при получении функции зависимости количества выполняемых инструкций от размера задачи

|  |  |  |
| --- | --- | --- |
| Номер инструкции | Код алгоритма addElement | Количество операций в инструкции |
|  | temp2←cin | 2 |
|  | element←cin | 2 |
|  | For pos←0 to n do | 1+3(n+1) |
|  | For i←10; temp !=0; i \*= 10 do | n(1+3(n+1)) |
|  | temp1 ← temp2 % i |  |
|  | if ((A[pos] % temp1) != 0) |  |
|  | break; |  |
|  | if ((A[pos] % temp1 == 0) && (temp1 < 10)) |  |
|  | addPos(pos + 1, element) | (14+10n-10pos) |
|  | return pos+1 |  |
|  | endif | - |

T(n) = 8+7n++-10pos

Асимптотическая временная сложность квадратичная – O().

Таблица 7. Форма представления операции 7 при получении функции зависимости количества выполняемых инструкций от размера задачи

|  |  |  |
| --- | --- | --- |
| Номер инструкции | Код алгоритма deleteKrat3 | Количество операций в инструкции |
|  | For pos←0 to n do | 1+3(n+1) |
|  | if (A[pos] % 3 == 0) | 3n |
|  | deletePos(pos) | 6+6n-6pos |
|  | endif | - |
|  | od | - |

T(n) = 10+12n-6pos

Асимптотическая временная сложность линейная – O(n-pos).

## 2.3. Код алгоритма на языке С++.

|  |
| --- |
| #include <cstdlib>  #include <iostream>  #include <ctime>  using namespace std;  typedef unsigned int typesize;  typedef int typeitem;  struct MyStruct  {  static const typesize N = 100;  typesize n = 0;  typeitem A[N] = {};  };  //заполнение структуры данных значениями  void create(MyStruct& s, typesize n, int temp);  //вывод структуры в консоль  void print(MyStruct s);  //удалить элемент в заданной позиции  int deletePos(MyStruct& s, typesize pos);  //вставить элемент в заданную позицию  int addPos(MyStruct& s, typesize pos, int element);  //Найти позицию элемента массива значение которого делится на каждую из цифр числа\*  int findElement(MyStruct& s, int temp2);  //Вставить в массив новый элемент после элемента, значение которого делится на каждую цифру значения  int addElement(MyStruct& s, int temp2, int element);  //Удалить из массива все элементы, кратные трем  void deleteKrat3(MyStruct& s);  int main()  {  setlocale(LC\_ALL, "Rus");  MyStruct A;  int n;  cout << "n = ";  cin >> n;  if (n<0 || n>A.N)  {  cout << "Вы ввели не правильное значение n";  return 1;  }    create(A, n, 1);  print(A);  deletePos(A, 5);  print(A);  addPos(A, 5, 888);  print(A);  findElement(A,14);  addElement(A, 123456789,3);  deleteKrat3(A);  print(A);  }  void create(MyStruct& s, typesize n, int temp)  {  typesize temp1;  if (temp == 0)  {  for (typesize i = 0; i < n; i++)  {  cout << "Введите значение " << i << endl;  cin >> temp1;  s.A[i] = temp1;  }  for (typesize i = n; i < s.N; i++)  s.A[i] = 0;  }  else if (temp == 1)  {  for (typesize i = 0; i < n; i++)  s.A[i] = rand();  }  else  {  cout << "Вы ввели не правильное значение нужно 0 или 1" << endl;  return;  }  s.n = n;  }  void print(MyStruct s)  {  cout << "Множество чисел \n";  for (typesize pos = 0; pos < s.n; pos++)  {  cout << "Элемент " << pos << " = " << s.A[pos] << endl;  }  }  int deletePos(MyStruct& s , typesize pos)  {  if (pos <= s.n && pos >= 0)  {  for (typesize i = pos; i < s.n; i++)  {  s.A[i] = s.A[i + 1];  }  s.n--;  return 0;  }  else  {  cout << "нет заданной позиции" << endl;  return -1;  }  }  int addPos(MyStruct& s, typesize pos, int element)  {  if (pos <= s.n && pos >= 0)  {  s.n++;  if (s.n > s.N)  {  cout << "n массив выходит за область допустимых значений" << endl;  s.n--;  return -1;  }  int temp2;  int temp1 = s.A[pos];  s.A[pos] = element;  for (int i = pos + 1; i < s.n; i++)  {  temp2 = s.A[i];  s.A[i] = temp1;  temp1 = s.A[i + 1];  temp1 = temp2;  }  return 0;  }  else  {  cout << "нет заданной позиции" << endl;  return -1;  }  }  int findElement(MyStruct& s,int temp2)  {  int temp1 = 1;  int t;  for (int pos = 0; pos < s.n; pos++)  {  t = temp2;  for (int i = 10; t != 0; i \*= 10)  {  temp1 = t % 10;  if ((s.A[pos] % temp1) != 0)  {  break;  }  if ((s.A[pos] % temp1 == 0) && (t < 10))  return pos;  t = temp2 / i;  }  }  return -1;  }  int addElement(MyStruct& s,int temp2, int element)  {  int temp1 = 1;  int t;  for (int pos = 0; pos < s.n; pos++)  {  t = temp2;  for (int i = 10; t != 0; i \*= 10)  {  temp1 = t % 10;  if ((s.A[pos] % temp1) != 0)  {  break;  }  if ((s.A[pos] % temp1 == 0) && (t < 10))  {  addPos(s,pos + 1, element);  return pos + 1;  }  t = temp2 / i;  }  }  return -1;  }  void deleteKrat3(MyStruct& s)  {  for (int pos = 0; pos < s.n; pos++)  {  if (s.A[pos] % 3 == 0)  deletePos(s,pos);  }  } |

## 2.4. Тестирование алгоритма.

Составленные тесты (Таблица 2), которые обрабатывают базовые и крайние случаи для размерности массива равной 20 (n = 20).

Таблица 8 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| заполнение структуры данных значениями void create(int temp) | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | n = 20  create(X, n, 1) | Элемент 0 = 41  Элемент 1 = 18467  Элемент 2 = 6334  Элемент 3 = 26500  Элемент 4 = 19169  Элемент 5 = 15724  Элемент 6 = 11478  Элемент 7 = 29358  Элемент 8 = 26962  Элемент 9 = 24464  Элемент 10 = 5705  Элемент 11 = 28145  Элемент 12 = 23281  Элемент 13 = 16827  Элемент 14 = 9961  Элемент 15 = 491  Элемент 16 = 2995  Элемент 17 = 11942  Элемент 18 = 4827  Элемент 19 = 5436 |
| 2 | n = -12  create(X, n, 1) | Вы ввели не правильное значение n |

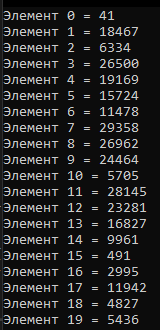




Рисунок 1 - Скриншоты тестирований

Таблица 9 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| вывод структуры в консоль void print() | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | print(X) | Элемент 0 = -858993460  Элемент 1 = -858993460  Элемент 2 = -858993460  Элемент 3 = -858993460  Элемент 4 = -858993460  Элемент 5 = -858993460  Элемент 6 = -858993460  Элемент 7 = -858993460  Элемент 8 = -858993460  Элемент 9 = -858993460  Элемент 10 = -858993460  Элемент 11 = -858993460  Элемент 12 = -858993460  Элемент 13 = -858993460  Элемент 14 = -858993460  Элемент 15 = -858993460  Элемент 16 = -858993460  Элемент 17 = -858993460  Элемент 18 = -858993460  Элемент 19 = -858993460 |

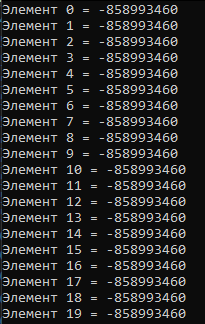


Рисунок 2 - Скриншоты тестирований

Таблица 10 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| удалить элемент в заданной позиции int deletePos(int pos) | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | n = 20  create(X, n, 1)  deletePos(X, 0) | Элемент 0 = 18467  Элемент 1 = 6334  Элемент 2 = 26500  Элемент 3 = 19169  Элемент 4 = 15724  Элемент 5 = 11478  Элемент 6 = 29358  Элемент 7 = 26962  Элемент 8 = 24464  Элемент 9 = 5705  Элемент 10 = 28145  Элемент 11 = 23281  Элемент 12 = 16827  Элемент 13 = 9961  Элемент 14 = 491  Элемент 15 = 2995  Элемент 16 = 11942  Элемент 17 = 4827  Элемент 18 = 5436 |
| 2 | n = 20  create(X, n, 1)  deletePos(X, -1) | нет заданной позиции |

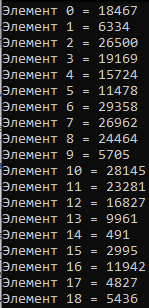
 

Рисунок 3 - Скриншоты тестирований

Таблица 11 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| вставить элемент в заданную позицию int addPos(int pos, int element) | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | n = 20  create(X, n, 1)  deletePos(X, 0)  addPos(X, 1, 228); | Элемент 0 = 41  Элемент 1 = 228  Элемент 2 = 18467  Элемент 3 = 6334  Элемент 4 = 26500  Элемент 5 = 19169  Элемент 6 = 15724  Элемент 7 = 11478  Элемент 8 = 29358  Элемент 9 = 26962  Элемент 10 = 24464  Элемент 11 = 5705  Элемент 12 = 28145  Элемент 13 = 23281  Элемент 14 = 16827  Элемент 15 = 9961  Элемент 16 = 491  Элемент 17 = 2995  Элемент 18 = 11942  Элемент 19 = 4827  Элемент 20 = 5436 |
| 2 | n = 20  create(X, n, 1)  deletePos(X, 0)  addPos(X, -1, 228); | нет заданной позиции |

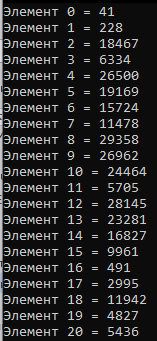


Рисунок 4 - Скриншоты тестирований

Таблица 12 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| Найти позицию элемента массива значение которого делится на каждую из цифр числа.\* int findElement(int temp2) | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | n = 20  create(X, n, 1)  findElement(X, 45)  print(X) | Элемент 0 = 41  Элемент 1 = 18467  Элемент 2 = 6334  Элемент 3 = 26500  Элемент 4 = 19169  Элемент 5 = 15724  Элемент 6 = 11478  Элемент 7 = 29358  Элемент 8 = 26962  Элемент 9 = 24464  Элемент 10 = 5705  Элемент 11 = 28145  Элемент 12 = 23281  Элемент 13 = 16827  Элемент 14 = 9961  Элемент 15 = 491  Элемент 16 = 2995  Элемент 17 = 11942  Элемент 18 = 4827  Элемент 19 = 5436  возвращаемое значение 3 |
| 2 | n = 20  create(X, n, 1)  findElement(X, 123456789)  print(X) | Элемент 0 = 41  Элемент 1 = 18467  Элемент 2 = 6334  Элемент 3 = 26500  Элемент 4 = 19169  Элемент 5 = 15724  Элемент 6 = 11478  Элемент 7 = 29358  Элемент 8 = 26962  Элемент 9 = 24464  Элемент 10 = 5705  Элемент 11 = 28145  Элемент 12 = 23281  Элемент 13 = 16827  Элемент 14 = 9961  Элемент 15 = 491  Элемент 16 = 2995  Элемент 17 = 11942  Элемент 18 = 4827  Элемент 19 = 5436  возвращаемое значение -1 |

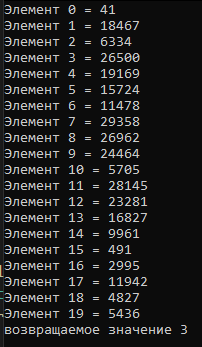
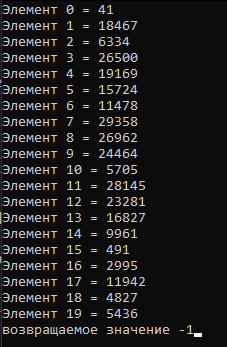
 

Рисунок 5 - Скриншоты тестирований

Таблица 13 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| Вставить в массив новый элемент после элемента, значение которого делится на каждую цифру значения. int addElement(int temp2, int element) | | |
| Номер теста |  | Эталон результата |
| 1 | n = 20  create(X, n, 1)  addElement(X, 45, 3); | Элемент 0 = 41  Элемент 1 = 18467  Элемент 2 = 6334  Элемент 3 = 26500  Элемент 4 = 3  Элемент 5 = 19169  Элемент 6 = 15724  Элемент 7 = 11478  Элемент 8 = 29358  Элемент 9 = 26962  Элемент 10 = 24464  Элемент 11 = 5705  Элемент 12 = 28145  Элемент 13 = 23281  Элемент 14 = 16827  Элемент 15 = 9961  Элемент 16 = 491  Элемент 17 = 2995  Элемент 18 = 11942  Элемент 19 = 4827  Элемент 20 = 5436 |
| 2 | n = 20  create(X, n, 1)  addElement(X, 123456789, 3); | возвращаемое значение -1 |

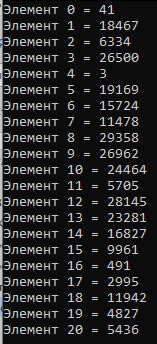


Рисунок 6 - Скриншоты тестирований

Таблица 14 - Таблица тестирования

|  |  |  |
| --- | --- | --- |
| Удалить из массива все элементы, кратные трем void deleteKrat3() | | |
| Номер теста | Входные данные | Эталон результата |
| 1 | n = 20  create(X, n, 1)  deleteKrat3(X); | Элемент 0 = 41  Элемент 1 = 18467  Элемент 2 = 6334  Элемент 3 = 26500  Элемент 4 = 19169  Элемент 5 = 15724  Элемент 6 = 29358  Элемент 7 = 26962  Элемент 8 = 24464  Элемент 9 = 5705  Элемент 10 = 28145  Элемент 11 = 23281  Элемент 12 = 9961  Элемент 13 = 491  Элемент 14 = 2995  Элемент 15 = 11942  Элемент 16 = 5436 |

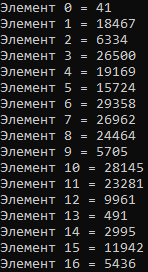


Рисунок 7 - Скриншоты тестирований

# Выводы

В ходе самостоятельной работы была изучена технология разработки алгоритмов на основе абстрактных типов данных. Были приобретены умения и навыки в определении и реализации абстрактного типа данных задачи на одномерном статическом массиве, который является одной из наиболее эффективных структур представления данных в программе.

Также были приобретены навыки по реализации алгоритмов операций над массивом через аппарат функций языка С++. Были изучены операции модификации массива, такие как вставка и удаление элементов, а также другие операции обработки массива как структуры данных.

В результате работы было достигнуто основная цель - освоение технологии разработки программ на основе абстрактного типа данных и приобретение необходимых навыков для его реализации на одномерном статическом массиве.

# 4. Информационные источники

1. Приложение к практическим работам – СДО (online-edu.mirea.ru)

2. Структуры и алгоритмы обработки данных – методические указания / Скворцова Л.А., Филатов А.С., Гусев К.В., Трушин С.М. - Москва, МИРЭА - Российский технологический университет, 2023.