Структура одномерный массив

Тема. Стандартные типы данных языка программирования для представления многоэлементных однородных структур данных задачи в программе

Цель.

- 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

Оглавление

[Задание 1. Разработать программу для выполнения операций на статическом массиве в отдельном проекте решения 1](#_Toc95055668)

[Задание 2. Разработать программу для выполнения всех операций варианта над динамическим массивом 2](#_Toc95055669)

[Задание 3. Разработать программу для выполнения всех операций варианта над динамическим массивом, используя контейнер <vector> для его представляя в программе. 2](#_Toc95055670)

[Задание 4. Подготовить отчет по выполнению заданий 1 и 2 по статическому и динамическому массивам. По заданию 3 в отчет включить код, комментируя функции. 2](#_Toc95055671)

[Структура отчета должен включать 2](#_Toc95055672)

[Варианты 2](#_Toc95055673)

[Пример разработки приложения и оформления отчета по заданию 1 5](#_Toc95055674)

[1) Заполнение исходного массива значениями с клавиатуры 6](#_Toc95055675)

[2) Вывод значений массива 6](#_Toc95055676)

[3) Функции декомпозиции 6](#_Toc95055677)

[Пример разработки приложения и оформления отчета по заданию 2 8](#_Toc95055678)

[Статический массив 11](#_Toc95055679)

[Динамический массив 13](#_Toc95055680)

[Контейнер <vector> - современная реализация динамических массивов 15](#_Toc95055681)

# Задание 1. Разработать программу для выполнения операций на статическом массиве в отдельном проекте решения

1. Разработать функции для заполнения и отображения значений массива.
2. Выполнить декомпозицию задач варианта (смотри пример оформления отчета).
3. Выполнить разработку задачи варианта, отмеченной символом \*.
4. Постановка задачи
5. Модель решения

Выполнить декомпозицию задачи. Описать ваш подход к решению каждой выявленной подзадачи (может быть и одна подзадача),

Если решение задачи можно описать на языке математике, то приведите математический аппарат ее решения.

Приведите прототипы функций каждой подзадачи и опишите постусловие и постусловие.

1. Разработать алгоритм для каждой подзадачи и алгоритм основной задачи.
2. Реализовать функции подзадач и основной функции.
3. Отладить каждую функцию на успешных и безуспешных тестах.

Примечание. Выполняйте отладку функций последовательно по одной, сначала те, которые вызываются другой функцией. Только убедившись в том, что функция отлажена, переходите к отладке (можно к разработке) той, которая вызывает.

1. Для других двух задач варианта выполнить: постановку задачи, описать модель решения (как в 2) п.3, подготовить тесты.
2. Разработать основную программу, которая должна продемонстрировать работу всех операций над массивом.
3. Отладить программу. Привести в конце модуля программы тесты.

# Задание 2. Разработать программу для выполнения всех операций варианта над динамическим массивом

В отдельном проекте решения, созданного в задании 1. Скопируйте в проект функции из проекта задания 1, внесите изменения в функции, изменяющие размер массива.

# Задание 3. Разработать программу для выполнения всех операций варианта над динамическим массивом, используя контейнер <vector> для его представляя в программе.

# Задание 4. Подготовить отчет по выполнению заданий 1 и 2 по статическому и динамическому массивам. По заданию 3 в отчет включить код, комментируя функции.

# Структура отчета

1. Условие задания 1
   1. Задачи варианта
   2. Разработка задачи, отмеченной \*, т.е. всю разработку от 1) до 5)
   3. Разработка двух других задач и основной программы
   4. Код реализации программы
   5. Результаты тестирования
2. Условие задания 2
   1. Код реализации
   2. Тесты
   3. Результаты тестирования

Пример оформления отчета приведен в конце данного документа.

# Варианты

|  |  |
| --- | --- |
| № | Задачи варианта. Операции над элементами структуры |
| 1 | 1. Найти индекс элемента массива, являющегося простым числом.\* 2. Вставить новый элемент в массив в позицию, следующую за первым простым числом в массиве. 3. Удалить каждый элемент массива, который кратен 7. |
| 2 | 1. Вставить новое значение в массив перед первым элементом массива.\* 2. Определить, образуют ли числа массива арифметическую прогрессию. 3. Удалить элементы массива, в значениях которых первая и последняя цифры одинаковы. |
| 3 | 1. Вставить новый элемент в массив перед элементом в заданной позиции. 2. Найти последнее вхождение в массив числа, у которого равны первая и последняя цифры. 3. Удалить элементы массива кратные 5. |
| 4 | 1. Вставить новое значение после значения в заданной позиции. 2. Определить, сколько раз входит в массив максимальное значение массива (одним алгоритмом). 3. Удалить все числа массива, которые являются совершенными числами (число равно сумме своих делителей кроме самого числа: 6, 28). |
| 5 | 1. Вставить новый элемент в массив перед элементом, у которого четное количество цифр. 2. Удалить все четные числа массива. 3. Найти максимальное число среди элементов массива, расположенных на четных местах. |
| 6 | 1. Найти индекс элемента массива цифры которого упорядочены по возрастанию. Считать, что такое число одно. 2. Вставить новый элемент после элемента, цифры которого упорядочены по возрастанию. 3. Удалить число, которое расположено перед числом, цифры которого упорядочены по возрастанию. |
| 7 | 1. Найти индекс элемента массива, цифровой корень которого равен 7.   Подсказка. Цифровой корень – это однозначное число. Алгоритм определения цифрового корня: дано число 277, сумма его цифр 16 – двухзначное; снова сумма но уже 16 равна 7 – уже однозначное – это цифровой корень числа 277.   1. Вставить новый элемент перед элементом, цифровой корень которого равен 7. Считать, что такое число одно. 2. Удалить элементы массива цифровой корень которых равен 7. |
| 8 | 1. Найти индекс элемента массива, наибольшая цифра значения которого – это первая цифра числа. 2. Вставить новый элемент в массив перед элементом, наибольшая цифра значения которого – это первая цифра числа. 3. Удалить элементы массива, наибольшая цифра значения которых – это первая цифра числа. |
| 9 | 1. Найти индекс элемента массива (первое вхождение), которое является палиндромом. 2. Удалить элементы массива, расположенное непосредственно перед элементом, содержащим число палиндром. 3. Вставить новый элемент в массив после элемента массива, который является палиндромом. |
| 10 | 1. Найти индекс элемента массива (первое вхождение), которое является совершенным (число равно сумме своих делителей кроме самого числа: 6, 28). 2. Вставить новый элемент в массив после элемента, который является совершенным. 3. Удалить элемент массива, расположенный перед элементом, содержащим совершенное число. |
| 11 | 1. Найти индекс элемента массива, цифры которого (слева направо) образуют последовательность Фибоначчи. 2. Вставить новый элемент в массив после элемента, цифры которого образуют последовательность чисел Фибоначчи. 3. Удалить элемент массива, расположенный перед элементом, цифры которого образуют последовательность чисел Фибоначчи. |
| 12 | 1. Найти индекс максимального элемента массива, среди четных чисел массива. 2. Вставить новый элемент в массив после элемента с максимальным значением среди черных чисел массива. 3. Удалить элемент массива, расположенный перед элементом, с максимальным значением среди черных чисел массива. |
| 13 | 1. Найти максимальное значение массива. 2. Вставить максимальное значение массива после элемента, у которого первая и последняя цифры равны. 3. Удалить элементы массива, цифры которых образуют последовательность чисел Фибоначчи, в которой первое и второе число равно 1. |
| 14 | 1. Найти индекс первого вхождения минимального значения среди отрицательных чисел массива. 2. Вставить новый элемент массива после минимального элемента массива. 3. Удалить все элементы массива равные минимальному значению в массиве среди отрицательных чисел. |
| 15 | 1. Найти индекс элемента (первое вхождение) массива, у которого все цифры одинаковые. 2. Вставить новый элемент в массив после элемента, все у которого все цифры одинаковые. 3. Удалить элементы, у которого все цифры одинаковые. 4. Определить, упорядочен ли массив по возрастанию или по убыванию. |
| 16 | 1. Найти индекс элемента массива (первое вхождение) двоичный код значения которого содержит ровно три единицы. 2. Вставить новый элемент в массив после элемента, двоичный код значения которого содержит ровно три единицы. 3. Удалить элементы массива, двоичный код значения которых содержит ровно три единицы. |
| 17 | 1. Найти индекс элемента массива, старшая цифра значения которого равна заданной. 2. Вставить новый элемент в массив перед элементом массива, старшая цифра значения которого равна заданной 3. Удалить все элементы массива, старшая цифра значений которых равна заданной. |
| 18 | * + - 1. Найти индекс элемента массива (первое вхождение) троичный код значения которого содержит ровно две двойки.       2. Вставить новый элемент в массив после элемента, троичный код значения которого содержит ровно две двойки.       3. Удалить элементы массива, троичный код значений которых содержит ровно две двойки. |
| 19 | 1. Найти индекс элемента массива, значение которого содержит цифру 0. 2. Вставить новый элемент в массив после элемента, значение которого не содержит цифру 0. 3. Удалить элементы массива, значение которого содержит цифру 0. |
| 20 | * + - 1. Найти индекс элемента массива, произведение цифр которого больше нуля и кратно трем.       2. Вставить новый элемент в массив перед элементом с максимальным значением.       3. Удалить элемент массива, произведение цифр которого больше нуля и кратно трем. |
| 21 | 1. Найти индекс элемента массива, сумма цифр значения которого кратна 7. 2. Вставить новый элемент в массив перед минимальным элементом, сумма цифр значения которого кратна 7. 3. Удалить элементы массива, сумма цифр значения которого кратна 7. |
| 22 | 1. Найти индекс 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. |
| 23 | 1. Определить, упорядочены ли значения в массиве по возрастанию. 2. Если значения в массиве упорядочены по возрастанию, то удалить из массива элементы, которые кратны введенному значению. 3. Если значения в массиве не упорядочены по возрастанию, то вставить новый элемент в массив перед первым элементом. |
| 24 | 1. Найти индексы (начальный и конечный) самой длинной, упорядоченной по возрастанию подпоследовательности (части массива). 2. Вставить новый элемент перед элементом, с начальным индексом подпоследовательности. 3. Удалить все элементы найденной подпоследовательности. |
| 25 | 1. Определить, сколько раз в массиве встречается максимальное значение и сформировать массив индексов этих элементов. 2. Удалить все максимальные значения, используя массив их индексов. 3. Если в массиве только одно максимальное значение, то добавить такое же значение в массив. |
| 26 | 1. Определить, упорядочены ли значения в массиве по возрастанию. 2. Если значения в массиве не упорядочены по возрастанию, то удалить из массива элементы, которые кратны введенному значению. 3. Если значения в массиве упорядочены по возрастанию, то вставить новый элемент в массив перед элементом с большим его по значению. |
| 27 | 1. Определить, упорядочены ли значения в массиве по убыванию. 2. Если значения в массиве не упорядочены по убыванию, то удалить из массива элементы, значения которых содержат цифру 5. 3. Если значения в массиве упорядочены по убыванию, то вставить новый элемент в массив перед элементом с меньшим его по значению. |
| 28 | 1. Сформировать новый массив из простых чисел исходного массива, вставляя каждое значение (кроме первого значения) так, чтобы числа образовали в результате возрастающую последовательность. 2. Удалить минимальное число нового массива. 3. Определить у скольких чисел исходного массива количество делителей больше трех. |
| 29 | 1. Сформировать новый массив из чисел исходного массива, сумма цифр которых кратна 7, вставляя каждое значение (кроме первого значения) так, чтобы числа образовали в результате убывающую последовательность. 2. Удалить минимальное число нового массива. 3. Определить у скольких чисел исходного массива цифры образуют возрастающую последовательность. |
| 30 | Определить сколько в массиве простых чисел Мерсенна. Считать натуральное число M простым числом Мерсенна, если оно удовлетворяет свойствам: 1) М – простое число 2) число М+1 является степенью двойки. Например, число М=31.  Удалить минимальное число, которое является степенью степень числа 2.  Вставить в массив новый элемент после элемента значение которого является максимальным простым числом Мерсенна. |

# Пример разработки приложения и оформления отчета по заданию 1

1. Условие задачи варианта
   1. Скопировать задание
   2. Скопировать условие варианта

Дан массив из n элементов целого типа long.

Сформировать массив из чисел Армстронга (153=13+53+33)

1. Разработка программы
   1. Постановка задачи

Дано. Дан массив из n элементов целого типа long.

Результат. Сформировать массив из чисел Армстронга (153=13+53+33)

Ограничения. Массив натуральных чисел

* 1. Описание модели решения

Исходный массив А статический максимального размера N=100.

Текущий размер n массива А определяет пользователь n<=N.

Новый массив B, который формируется программой – статический максимального размера N, фактический размер массива nB будет определен программой.

Математическая модель определения числа Армстронга

Пусть Х исходное число, проверяемое на число Армстронга

Степень равна количеству цифр в числе -p .

Найти сумму возведенных в степень p цифр числа Х это S и сравнить исходное число Х с полученной суммой S. Если X=S то нашли число Армстронга.

Модель формирования массива чисел Армстронга

nB – индекс элемента в который вставляется значение и текущий размер массива В.

Каждое найденное число вставляется в массив В по индексу nB и после вставки nB увеличивается.

* 1. Декомпозиция – список алгоритмов, которые требуется разработать в соответствии с исследованной моделью
     1. Список подзадач

1. Определение количества цифр в числе
2. Возведение целого числа в степень
3. Определение числа Армстронга
4. Формирование нового массива из чисел исходного
   * 1. Определение прототипов функций
5. Заполнение исходного массива значениями с клавиатуры

Предусловие. n – число заполняемых элементов,0 ≤n≤Max, где MAX – максимальное число элементов, L – массив с переменной верхней границей.

Постусловие. Заполненный массив из n элементов

void input\_ar(Array\_Static L[], int n);

1. Вывод значений массива

Предусловие. n>0

Постусловие. Вывод значений массива

void output\_ar(Array\_Static L[], int n);

1. Функции декомпозиции

***//Определение количества цифр в числе***

Предуслвие. x≥10

Постусловие. Результат целое, сумма цифр

int count(long x);

***//Возведение целого числа в степень p***

Предуслвие. а>0 – цифра числа, p>0 – количество цифр в числе

Постусловие. Результат целое, возведение а в степень p

long double pow\_1(unsigned short a, unsigned short p);

***//Определение числа Армстронга***

Предуслвие. х>0 целое число,

Постусловие. True если х число Армстронга, false иначе

bool Armstrong(long x);

***//Формирование массива чисел Армстронга***

Предусловие. A исходный массив размера n>0. В – указатель на массив, подготовленный в функции main под результат.

Постусловие. Результат массив В из nB элементов. Если в массиве А нет чисел Армстронга, то nB=0 и массив В пустой.

void newArrayB(long \*A, int n, long \*B, int &nB);

* 1. Разработка алгоритмов операций и представление его на псевдокоде

1. Алгоритмы вода и вывода массива определим при реализации
2. Алгоритмы задач декомпозиции

Алгоритм функции Armstrong(long x);

Armstrong(long x){

int p←count(x); //количество цифр в числе и значение степени

int sum←0;

long copyx←x; //копия исходного числа

while(x!=0)

do

sum←sum+pow\_1(x%10,p);

x=x/10;

od

if (copyx=sum)

then результат true

else результат false

}

Далее алгоритмы других функций

* 1. Набор тестов для тестирования программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер теста | Исходные данные | Ожидаемый результат | Результат программы | Тест пройден/не пройден |
| 1 | n=3  А(1, 2, 3) | nB=3  B(1,2,3) |  |  |
| 2 | n=5  А(11, 12, 13,14, 15) | nB=5  B() Нет таких чисел |  |  |
| 3 | nB=3  B(1,22,153) | nB=2  B(1,153) |  |  |

1. **Код программы**
2. **Таблица тестов для каждой функции и программы.**

# Пример разработки приложения и оформления отчета по заданию 2

1. Условие задачи варианта
   1. Скопировать задание
   2. Скопировать условие варианта

Дан массив из n элементов целого типа long. Массив исходный динамический.

Сформировать массив из чисел Армстронга (153=13+53+33). Массив динамический.

1. Определение функций

Создание массива из n элементов целого типа для исходного массива.

Предусловие Нет

Постусловие вводит размер массива с клавиатуры. Создает динамический массив из n элементов целого типа. Возвращает указатель на массив из n и n, либо NULL, если память под массив не выделена.

long \*create\_array(int &n);

Добавление нового элемента в массив

Предусловие. Массив должен существовать, т.е. L отлично от NULL.

Постусловие. Увеличивает размер массива на один элемент, добавляя новую ячейку в конец массива. Используется функция realloc из malloc.h

void insert\_elems(long \*&L,int &n,long x);

Удаление последней ячейки из массива

Предусловие. Массив должен существовать.

Постусловие. Из массива удаляется последняя ячейка. Размер массива уменьшается.

void delete\_elems(long \*&L,int &n);

Ввод значений

Предусловие. n – число заполняемых элементов,0 ≤n≤Max, где MAX – максимальное число элементов, L – массив с переменной верхней границей.

Постусловие. Заполненный массив из n элементов

void input\_ar(long L[], int n);

Вывод значений массива

void output\_ar(long L[], int n);

Предусловие. n>0

Постусловие. Вывод значений массива

*//Операции для решения поставленной задачи*

Определение количества цифр в числе

Предуслвие. n≥10

Постусловие. Результат целое, сумма цифр

int count(long n);

Возведение целого числа в степень

Предуслвие. а>0 – цифра числа, p>0 – количество цифр в числе

Постусловие. Результат целое, возведение а в сtтепень p

long double pow\_1(unsigned short a, unsigned short p);

Определение числа Армстронга

Предуслвие. x>0 целое число,

Постусловие. Результат true если х число Армстронга и false иначе

bool Armstrong(long x);

Формирование массива чисел Армстронга

Предусловие. A исходный массив размера n>0.

Постусловие. Результат: создает и заполняет массив B если в массиве А есть числа Армстронга; изменяет nB если числа Армстронга есть в А. Если в массиве А нет чисел Армстронга, то nB=0 и результат возвращаемый функцией NULL

long\* newArrayB(long \*A, int n, int &nB);

1. **Реализация функций приложения**

#include <iostream>

#include"malloc.h"

#include"string.h"

long \*create\_array(int &n);

void output\_ar(long L[], int n);

void input\_ar(long L[],int n);

void insert\_elems(long \*&L,int &n,long x);

void delete\_elems(long \*&L,int &n);

void insert(long \*&L,int &n,long x);

long\* newArrayB(long \*A, int n, int &nB);

***//Определение количества цифр в числе***

int count(long x);

***//Возведение целого числа в степень p***

long double pow\_1(unsigned short a, unsigned short p);

***//Определение числа Армстронга***

bool Armstrong(long x);

int main(int argc, char\* argv[])

{int n;

long \*A=create\_array(n);

input\_ar(A, n);

int x=10;

insert\_elems( A, n,x); //добавили еще одно значение в массив А

output\_ar(A, n);

delete\_elems(A,n); //удалили последний элемент

output\_ar(A, n);

cout<<(sizeof(a)/sizeof(long))<<endl; //определили новый размер массива А

int nB=0;

long \*B=newArrayB(A, n, nB);

if (nB>0){

cout<<”Найдены числа Армстронга в массиве А\n”;

output\_ar(B, nB);

}

else

cout<<”Не найдены числа Армстронга в массиве А\n”;

return 0;

}

long \*create\_array(int &n)

{

cout<<"Введите количество элементов n=";

cin>>n;

long \*ptr=new long[n];

return ptr;

}

void input\_ar(long L[],int n)

{ cout<<"введите "<<n<<"элементов ";

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

cin>>L[i];

}

void output\_ar(long L[], int n)

{ cout<<"Массив "<< endl;

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

cout<<"a["<<i<<"]="<<L [i]<<endl;

}

void insert(long \*&L,int &n,long x) //вставка нового значения в массив

{

memcpy(L,L,n+1); //увеличение размера массива –функция из string.h

L[n]=x;n++;

}

void insert\_elems(long \*&L,int &n,long x) //вставка нового значения в массив

{

L=(long\*) realloc((long\*)L,(n+1)\*sizeof(long)); //увеличение размера массива –функция из malloc.h

//увеличение памяти по указателю L

L[n]=x; n++;

}

void delete\_elems(long \*&L,int &n) //удаление элемента из массива

{

L=(long\*)realloc((long\*)L,(n-1)\*sizeof(long)); // уменьшение размера памяти по указателю L

n--;

**}**

long\* newArrayB(long \*A, int n, int &nB){

long \*B=new long[0];

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

if (Armstrong(A[i]==true)

if (nB==0)

long \*B=new long[1];

else

B=(long\*)realloc((long\*)B,sizeof(long)\*(nB+1));

B[nB]=A[i];

nB++;

}

return (nB>0?B:NULL);

}

# Статический массив

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.

Формат определения одномерного статического массива

*[КП] спецификатор базового типа массива* Имя массива *[количество элементов]*

Где:

*Количество элементов –* константа или макроопределение

Примеры объявления массива

#define Len 100

float z[len];

int main()

{

int x[100];

double y[Len];

}

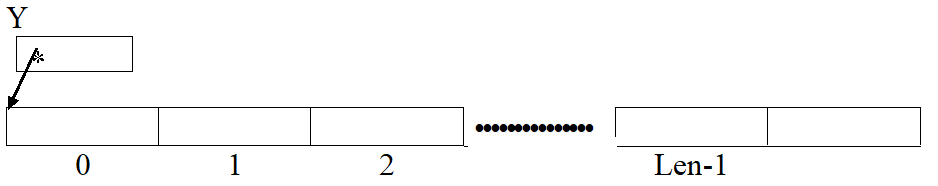
*Спецификатор базового типа массива* – это любой простой тип языка Си, включая указатель.

Представление в памяти массива из Len элементов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  | ∙∙∙∙∙∙∙∙∙∙∙∙∙∙∙ |  |
| 0 | 1 | 2 |  | Len-1 |

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. Это можно представить так:



Объем памяти выделенной переменной можно рассчитать по формуле Len\*объем одного элемента.

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, т.о. элемент массива – это отдельная переменная в составе массива.

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

Пример заполнения массива с переменной верхней границей при инициализации и при вводе с клавиатуры

int main()

{ int i ,y[4]={5, 6, 7,8}; //инициализация массива

// вывод массива

for(i=0;i<4;i++)

{

cout<< y[i]<<″ ″

}

//заполнение с клавиатуры

const int Len=100;

int x[Len], n;

cout<<″ Введите количество обрабатываемых элементов массива″;

cin>>n;

if(n>0 && n<Len)

{ cout<<″Введите ″<<n<<″чисел″);

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

{

cin>> x[i];

}

}

else

cout<<″ n должно быть больше нуля″;

}

Пример заполнения статического массива датчиком случайных чисел из диапазона от 100 до 300.

Примечание.

1)Функция rand() генерирует числа в диапазоне от 0 до RAND\_MAX. RAND\_MAX— это константа, определённая в библиотеке*cstdlib.h.* Для MVS RAND\_MAX= 32767,но оно может быть и больше, в зависимости от компилятора.

2) Формула генерации случайных чисел по заданному диапазону

*random\_number = firs\_value + rand() % last\_value;*

где

firs\_value - минимальное число из желаемого диапазона

last\_value - ширина выборки

3) Чтобы при различных запусках программы датчик формировал новое значение, необходимо настроить функцию rand(), передав ей новое начальное значение для алгоритма формирования случайного числа. Для этого надо в программу включить вызов функции *srand( time(0) );*из *time.h.* до вызова функции *rand().*

#include “stdio.h”

#include "stdlib.h"

#include "time.h"

#define Len 100

int main()

{

int x[Len], n;

cout<<″ Введите количество обрабатываемых элементов массива″;

cin>>n;

if(n>0 && n<Len)

{ *srand( time(0) );*

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

{

x[i]=100+rand()%300;

}

}

else

cout<<″n должно быть больше нуля″;

}}

# Динамический массив

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

Примечание. *Как же писать код программы и обращаться к ячейкам массива, если он еще не создан, т.е. нет переменной и соответственно память не выделена? Для этого в программе разработан тип данных – указатель. Это переменная, значением которой может быть адрес.*

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***.

**Функции языка Си по управлению динамическим массивом из модуля malloc.h:**

1. Выделение памяти под переменную в динамической памяти (создание динамической переменной)

void\* malloc(выражение);

*Выражение* определяет объем памяти в байтах.

Функция запрашивает у диспетчера динамической памяти (ДП) необходимое количество последовательно расположенных байтов, заданное выражением, если такой участок есть, то результат функции – адрес первого байта выделенной области, если такого свободного объема нет, то результат функции NULL.

Так как результат функции не типизированный указатель, то результат нужно привести к базовому типу массива (или типу переменной, под которую выделяется память).

Пример создания динамического массива

int mai()

{

int n;

cout<<”Введите количество элементов массива”;

int \*x=(int \*)malloc(n);

double \*y=( double \*)malloc(n);

}

1. Удаление динамического массива

После того, как необходимость в динамическом массиве в приложении отпала, его надо из ДП удалить, так как она автоматически не освобождается от использованных переменных, иначе эта память считается занятой и новым переменным может не хватить свободной памяти.

void free(указатель на динамическую память);

Пример удаления динамического массива после использования

int mai()

{

int n;

cout<<”Введите количество элементов массива”;

int \*x=(int \*)malloc(n);

double \*y=( double \*)malloc(n);

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

cin>>x[i];

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

cout<<x[i];

free(x);

}

1. Изменение размера динамического массива

void\* realloc(указатель, выражение);

где

*указатель* – это указатель на массив, размер которого надо изменить;

*выражение* – определяет новый размер массива в байтах.

Функция изменяет размер массива, добавляя новые байты в конец массива или удаляя последнее байты, сохраняя при этом значения остальных элементов массива.

Пример применения realloc для увеличения и уменьшения размера массива

int main()

{

int n=5;

int \*x=(int \*) malloc(n);

x[0]=1; x[1]=2;x[2]=3; x[3]=4; x[4]=5;

//увеличить массив на один элемент

realloc(x,sizeof(int)\*(n+1)); n++;

x[5]=6;

//увеличить массив на два элемента

realloc(x,sizeof(int)\*(n+2));

x[6]=7; x[7]=8;

//уменьшить массив на один элемент

n--;

realloc(x,sizeof(int)\*n);

free(x);

}

# 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).

Объём вектора и изменение размера

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.

Метод reserve() используется для предотвращения ненужного перераспределения памяти. После вызова reserve(n), объём вектора гарантированно будет не меньше n.

1. Формат определения вектора в программе: vector<Т>. Где Т тип элемента вектора

Примеры создания вектора его конструкторами

vector<int> myVector; // Пустой вектор из элементов типа int

vector<float> myVector(10); // Вектор из 10-и элементов типа float

vector<char> myVector(5, ' '); // Вектор, состоящий из 5-и пробелов

class myType {

...

};

int n = 10;

vector< myType > myVector(n); // Вектор из 10-и элементов пользовательского типа T

1. Доступ к элементам вектора

По соглашению [C](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8_(%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F)) и [C++](https://ru.wikipedia.org/wiki/C%2B%2B), первый элемент имеет индекс 0, последний — size() – 1.

1. Применение операции индексирования.

vector<int> myVector(10);

cin>>myVector[0];

i=0;

myVector[i]\*=2;

myVector[1]=5;

Примечание. Если i>=myVector.size() то результат операции не определен

1. Применение методов

*а)* Метод ***at*** - доступ к любому элементу по индексу

Формат метода at

[const] T& ***at***(int i); где Т тип элемента вектора

Примечание. Может вернуть исключение out\_of\_range

б) Метод v.front() - доступ к первому элементу вектора

[const] T& front() (int i); где Т тип элемента вектора

Примечание. Результат не определен если вектор пуст

в) Метод back() - доступ к последнему элементу вектора

[const] T& back () (int i); где Т тип элемента вектора

Примечание. Результат не определен если вектор пуст

Пример применения методов доступа

int main()

{

vector<int> myVector{ 1, 2, 3, 4, 5, 6 };

cout<<myVector.at(0)<<endl;

cout << myVector.front() << endl;

cout << myVector.back() << endl;

std::cout << "Hello World!\n";

}

Некоторые методы управления

Класс vector — это контейнер. Согласно стандарту C++, любой контейнер должен содержать методы begin(), end(), size(), max\_size(), empty(), и swap().

Ниже приведён краткий перечень доступных методов с описанием и указанием [сложности](https://ru.wikipedia.org/wiki/O-%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Метод | Описание | [Сложность](https://ru.wikipedia.org/wiki/O-%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F) |
| [Операторы](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) | vector::[operator=](http://ru.cppreference.com/w/cpp/container/vector/operator=) | Копирует значение одного вектора в другой. | O(n) |
| vector::[operator==](http://ru.cppreference.com/w/cpp/container/vector/operator==) | Сравнение двух векторов | O(n) |
| Доступ к элементам | vector::[at](http://ru.cppreference.com/w/cpp/container/vector/at) | Доступ к элементу с проверкой выхода за границу | O(1) |
| vector::[operator[]](http://ru.cppreference.com/w/cpp/container/vector/operator_at) | Доступ к определённому элементу | O(1) |
| vector::[front](http://ru.cppreference.com/w/cpp/container/vector/front) | Доступ к первому элементу | O(1) |
| vector::[back](http://ru.cppreference.com/w/cpp/container/vector/back) | Доступ к последнему элементу | O(1) |
| [Итераторы](https://ru.wikipedia.org/wiki/%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)) | vector::[begin](http://ru.cppreference.com/w/cpp/container/vector/begin) | Возвращает итератор на первый элемент вектора | O(1) |
| vector::[end](http://ru.cppreference.com/w/cpp/container/vector/end) | Возвращает итератор на место после последнего элемента вектора | O(1) |
| vector::[rbegin](http://ru.cppreference.com/w/cpp/container/vector/rbegin) | Возвращает reverse\_iterator на конец текущего вектора. | O(1) |
| vector::[rend](http://ru.cppreference.com/w/cpp/container/vector/rend) | Возвращает reverse\_iterator на начало вектора. | O(1) |
| Работа с размером вектора | vector::[empty](http://ru.cppreference.com/w/cpp/container/vector/empty) | Возвращает true, если вектор пуст | O(1) |
| vector::[size](http://ru.cppreference.com/w/cpp/container/vector/size) | Возвращает количество элементов в векторе | O(1) |
| vector::[max\_size](http://ru.cppreference.com/w/cpp/container/vector/max_size) | Возвращает максимально возможное количество элементов в векторе | O(1) |
| vector::[reserve](http://ru.cppreference.com/w/cpp/container/vector/reserve) | Устанавливает минимально возможное количество элементов в векторе | O(n) |
| vector::[capacity](http://ru.cppreference.com/w/cpp/container/vector/capacity) | Возвращает количество элементов, которое может содержать вектор до того, как ему потребуется выделить больше места. | O(1) |
| vector::[shrink\_to\_fit](http://ru.cppreference.com/w/cpp/container/vector/shrink_to_fit) | Уменьшает количество используемой памяти за счёт освобождения неиспользованной ([C++11](https://ru.wikipedia.org/wiki/C%2B%2B11)) | O(1) |
| Модификаторы | vector::[clear](http://ru.cppreference.com/w/cpp/container/vector/clear) | Удаляет все элементы вектора | O(n) |
| vector::[insert](http://ru.cppreference.com/w/cpp/container/vector/insert) | Вставка элементов в вектор | Вставка в конец, при условии, что память не будет перераспределяться — O(1), в произвольное место — O(n) |
| vector::[erase](http://ru.cppreference.com/w/cpp/container/vector/erase) | Удаляет указанные элементы вектора (один или несколько) | O(n) |
| vector::[push\_back](http://ru.cppreference.com/w/cpp/container/vector/push_back) | Вставка элемента в конец вектора | O(1) |
| vector::[pop\_back](http://ru.cppreference.com/w/cpp/container/vector/pop_back) | Удалить последний элемент вектора | O(1) |
| vector::[resize](http://ru.cppreference.com/w/cpp/container/vector/resize) | Изменяет размер вектора на заданную величину | O(n) |
| vector::[swap](http://ru.cppreference.com/w/cpp/container/vector/swap) | Обменять содержимое двух векторов | O(1) |
| Другие методы | vector::[assign](http://ru.cppreference.com/w/cpp/container/vector/assign) | Ассоциирует с вектором поданные значения | O(n), если установлен нужный размер вектора, O(n\*log(n)) при перераспределении памяти |
| vector::[get\_allocator](http://ru.cppreference.com/w/cpp/container/vector/get_allocator) | Возвращает [аллокатор](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%BB%D0%BE%D0%BA%D0%B0%D1%82%D0%BE%D1%80" \o "Аллокатор), используемый для выделения памяти | O(1) |

Пример вставки нового элемента в вектор в первую, последнюю и внутреннюю позицию (по индексу 3)

vector<float> price = { 78.56, 34.07, 23,45, 61.08, 29.3 };

//вставка значения как первого элемента вектора и запоминание итератора на него

auto iterator = price.insert(price.begin(), 42.67);

//результат: 42.67,78.56, 34.07, 23,45, 61.08, 29.3

//вставка с использованием итератора

price.insert(iterator, 30.76);

//результат: 30.76, 42.67,78.56, 34.07, 23,45, 61.08, 29.3

//Initialize an integer variable

int position = 3;

//Insert the number at the particular position

iterator = price.insert(price.begin() + position, 52.56);

//результат: 30.76, 42.67,78.56, 52.56, 34.07, 23,45, 61.08, 29.3

Пример передачи вектора в качестве параметра

Вывод значений вектора на экран

void print(vector<int> X){

for(int i=0;i<X.size();i++)

cout<<X[i]<<” “;

}

Пример функции, которая создает вектор из определенного количества элементов и возвращает его в качестве результата

vector<int> createVector(int n){

vector<int> Х(n);

return X;

}

int main(){

vector<int> B = createVector(2);

B[0] = 5; B[1] = 6;

}

# Приложение 1 Форма отчета

**Задание 1**

Условие задания

Дан статический массив из целых беззнаковых элементов.

* Вставить новый элемент перед каждым элементом, у которого четное количество цифр.
* Удалить все числа массива, следующие за простым числом.
* Найти максимальное число среди четных чисел массива.
* Для сортировки массива использовать алгоритм линейного выбора элемента.

Декомпозиция

Задачу следует разбить на следующие подзадачи:

* Проверка количества цифр в числе на четность
* Вставка нового элемента перед каждым, у которого четное количество цифр
* Проверка, является ли число простым
* Удаление все числа массива, следующие за простым числом
* Поиск максимального числа среди четных чисел массива
* Сортировка массива методом линейного выбора

Определение функций

1. Проверять, четное ли количество цифр в числе, будем с помощью функции

A

element – uns. long

Логическое знач

Входные данные

Результат

even

**bool even(unsigned long element)**

1. Вставлять новый элемент перед каждым, соответствующим условию, будем с помощью функции

A

Arr[] – uns. Long

newElement – uns.long

&n - int

Входные данные

Результат

insert\_s

**void insert\_s(unsigned long arr[], int &n, unsigned long newElement)**

1. Проверять, является ли число простым, будем с помощью функции

A

Element – uns.long

Логическое знач

Входные данные

Результат

Simple

bool simple(unsigned long element)

1. Удалять все числа массива, следующие за простым числом, будем с помощью функции

A

arr – uns. Lon\*g

&n - int

Входные данные

Результат

Del\_s

void del\_s(unsigned **long\* arr, int &n)**

1. Искать максимальное число среди четных чисел массива будем искать с помощью функции

A

arr – uns. Lon\*g

n - int

Unsigned long

Входные данные

Результат

maxEvenNum

unsigned long **maxEvenNum(unsigned long\* arr, int n)**

1. Для сортировки массива методом линейного выбора будем использовать функцию

A

Arr – uns. Lon\*g

n - int

Unsigned long\*

Входные данные

Результат

sort

unsigned long\* sort(unsigned long\* arr, int n)

**Реализация функций**

unsigned long\* sort(unsigned long\* arr, int n) { //сортировка массива методом прямого выбора

int imin;

for (int i = 0; i < n - 1; i++) {

imin = i;

for (int j = i + 1; j < n; j++) {

if (arr[j] <= arr[imin]) {

imin = j;

}

}

swap(arr[i], arr[imin]);

}

return arr;

}

unsigned long maxEvenNum(unsigned long\* arr, int n) { //максимальное четное число массива

int max, i = n - 1;

bool flag = false;

arr = sort(arr, n);

if (!n) {

return 1;

}

else {

while (!flag) {

if (!(arr[i] % 2)) {

max = arr[i];

flag = true;

}

else {

i--;

}

}

}

if (n < 1) {

return 1;

}

else if (flag) {

return max;

}

else

{

return 1;

}

}

bool even(unsigned long element) { //проверка эл-та на четность кол-ва цифр

int calc = 0;

while (element != 0) {

element /= 10;

calc++;

}

return (calc % 2) ? false : true;

}

void insert\_s(unsigned long arr[], int &n, unsigned long newElement) { //вставка нового элемента после каждого с четными цифрами

int calc = n;

for (int i = 0; i < calc; i++) {

if (even(arr[i])) {

calc++;

for (int j = calc; j > i; j--) {

arr[j] = arr[j - 1];

}

arr[i] = newElement;

i++;

}

}

n = calc;

}

bool simple(unsigned long element) { //простое ли число

if (element == 0 || element == 1) {

return false;

}

else if (element == 2) {

return true;

}

else {

int calc = 0;

for (int i = 2; i <= element / 2; i++) {

if (!(element % i)) {

calc++;

}

}

return (calc >= 1) ? false : true;

}

}

void del\_s(unsigned long\* arr, int &n) {//удалить элементы, следующие за простым числом

int calc = 0;

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

if (simple(arr[i])) {

for (int j = i + 1; j < n; j++) {

arr[j] = 0;

}

calc++;

break;

}

else {

calc++;

}

}

n = calc;

}

**Кодирование алгоритма программы**

// Sem3Lab02PROG.cpp : Defines the entry point for the console application.

#include "stdafx.h"

#include <iostream>

#include "malloc.h"

using namespace std;

void create(unsigned long\* arr, int n);

void show(unsigned long\* arr, int n);

unsigned long\* sort(unsigned long\* arr, int n);

unsigned long maxEvenNum(unsigned long\* arr, int n);

bool even(unsigned long element);

void insert\_s(unsigned long arr[], int &n, unsigned long newElement);

bool simple(unsigned long element);

void del\_s(unsigned long\* arr, int &n);

int main()

{

setlocale(0, "Russian");

//=====================================================//

int n = 0;

int answer1 = 100;

unsigned long array1[100];

while (answer1 != 0) {

system("cls");

cout << "Лабораторная работа №2 ИКБО-07-16 Шамрай К.К. Вариант 5" << endl << endl;

cout << "Задание 1" << endl;

cout << "Меню\n";

cout << "1) Заполнить массив\n";

cout << "2) Вывести массив\n";

cout << "3) Отсортировать массив методом простого выбора\n";

cout << "4) Вставить новый элемент перед каждым элементом, у к-го четное кол-во цифр\n";

cout << "5) Удалить все числа массива, следующие за простым числом\n";

cout << "6) Найти максимальное число среди четных чисел массива\n";

cout << "0) Выход\n";

cout << "Ваш выбор: ";

cin >> answer1;

system("cls");

cout << "Лабораторная работа №2 ИКБО-07-16 Шамрай К.К. Вариант 5" << endl << endl;

switch (answer1)

{

case 1: {

cout << "Введите кол-во элементов в массиве: ";

cin >> n;

cout << "Введите элементы массива: ";

create(array1, n);

system("pause");

break;

}

case 2: {

cout << "Вывод массива\n";

show(array1, n);

system("pause");

break;

}

case 3: {

cout << "Отсортированный массив\n";

show(sort(array1, n), n);

system("pause");

break;

}

case 4: {

unsigned long newEl = 0;

cout << "Введите новый элемент: ";

cin >> newEl;

insert\_s(array1, n, newEl);

show(array1, n);

system("pause");

break;

}

case 5: {

cout << "Массив с удаленными числами\n";

del\_s(array1, n);

show(array1, n);

system("pause");

break;

}

case 6: {

cout << "Максимальное четное число массива: ";

if (maxEvenNum(array1, n) == 1) {

cout << "Четных чисел в массиве нет\n";

}

else {

cout << maxEvenNum(array1, n) << "\n";

}

system("pause");

break;

}

default:

break;

}

}

system("pause");

return 0;

}

void create(unsigned long\* arr, int n) { //заполнение массива

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

cin >> arr[i];

}

}

void show(unsigned long\* arr, int n) { //вывод массива

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

cout << arr[i] << " ";

}

cout << endl;

}

**Задание 2**

Условие задания

Дан динамический массив из целых беззнаковых элементов.

* Вставить новый элемент перед каждым элементом, у которого четное количество цифр.
* Удалить все числа массива, следующие за простым числом.
* Найти максимальное число среди четных чисел массива.
* Для сортировки массива использовать алгоритм линейного выбора элемента.

Декомпозиция

Задачу следует разбить на следующие подзадачи:

* Проверка количества цифр в числе на четность
* Вставка нового элемента перед каждым, у которого четное количество цифр
* Проверка, является ли число простым
* Удаление все числа массива, следующие за простым числом
* Поиск максимального числа среди четных чисел массива
* Сортировка массива методом линейного выбора

Определение функций

unsigned long\* sort(unsigned long\* arr, int n) { //сортировка массива методом прямого выбора

int imin;

for (int i = 0; i < n - 1; i++) {

imin = i;

for (int j = i + 1; j < n; j++) {

if (arr[j] <= arr[imin]) {

imin = j;

}

}

swap(arr[i], arr[imin]);

}

return arr;

}

unsigned long maxEvenNum(unsigned long\* arr, int n) { //максимальное четное число массива

int max, i = n - 1;

bool flag = false;

arr = sort(arr, n);

if (!n) {

return 1;

}

else {

while (!flag) {

if (!(arr[i] % 2)) {

max = arr[i];

flag = true;

}

else {

i--;

}

}

}

if (n < 1) {

return 1;

}

else if (flag) {

return max;

}

else

{

return 1;

}

}

bool even(unsigned long element) { //проверка эл-та на четность кол-ва цифр

int calc = 0;

while (element != 0) {

element /= 10;

calc++;

}

return (calc % 2) ? false : true;

}

void insert\_d(unsigned long\* arr, int &n, unsigned long newElement) {

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

if (even(arr[i])) {

//realloc(arr, (n + 1) \* sizeof(unsigned long));

arr = (unsigned long\*)realloc(arr, (n + 1) \* sizeof(unsigned long));

n++;

for (int j = n; j > i; j--) {

arr[j] = arr[j - 1];

}

arr[i] = newElement;

i++;

}

}

}

bool simple(unsigned long element) { //простое ли число

if (element == 0 || element == 1) {

return false;

}

else if (element == 2) {

return true;

}

else {

int calc = 0;

for (int i = 2; i <= element / 2; i++) {

if (!(element % i)) {

calc++;

}

}

return (calc >= 1) ? false : true;

}

}

void del\_d(unsigned long\* arr, int &n) { //удалить элементы, следующие за простым числом

int calc = n;

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

if (simple(arr[i])) {

calc--;

//realloc(arr, (n - calc) \* sizeof(unsigned long));

arr = (unsigned long\*)realloc(arr, (n - calc) \* sizeof(unsigned long));

break;

}

else {

calc--;

}

}

n -= calc;

}

**Кодирование алгоритма программы**

// Sem3Lab02PROG.cpp : Defines the entry point for the console application.

#include "stdafx.h"

#include <iostream>

#include "malloc.h"

using namespace std;

void create(unsigned long\* arr, int n);

void show(unsigned long\* arr, int n);

unsigned long\* sort(unsigned long\* arr, int n);

unsigned long maxEvenNum(unsigned long\* arr, int n);

bool even(unsigned long element);

void insert\_d(unsigned long arr[], int &n, unsigned long newElement);

bool simple(unsigned long element);

void del\_d(unsigned long\* arr, int &n);

int main()

{

setlocale(0, "Russian");

//=====================================================//

unsigned long \*array2;

int answer2 = 100;

int calc = 0;

while (answer2 != 0) {

system("cls");

cout << "Лабораторная работа №2 ИКБО-07-16 Шамрай К.К. Вариант 5" << endl << endl;

cout << "Задание 2" << endl;

cout << "Меню\n";

cout << "1) Заполнить массив\n";

cout << "2) Вывести массив\n";

cout << "3) Отсортировать массив методом простого выбора\n";

cout << "4) Вставить новый элемент перед каждым элементом, у к-го четное кол-во цифр\n";

cout << "5) Удалить все числа массива, следующие за простым числом\n";

cout << "6) Найти максимальное число среди четных чисел массива\n";

cout << "0) Выход\n";

cout << "Ваш выбор: ";

cin >> answer2;

system("cls");

cout << "Лабораторная работа №2 ИКБО-07-16 Шамрай К.К. Вариант 5" << endl << endl;

switch (answer2)

{

case 1: {

cout << "Введите кол-во элементов в массиве: ";

cin >> calc;

array2 = new unsigned long[calc];

cout << "Введите элементы массива: ";

create(array2, calc);

system("pause");

break;

}

case 2: {

cout << "Вывод массива\n";

show(array2, calc);

system("pause");

break;

}

case 3: {

cout << "Отсортированный массив\n";

show(sort(array2, calc), calc);

system("pause");

break;

}

case 4: {

unsigned long newEl = 0;

cout << "Введите новый элемент: ";

cin >> newEl;

insert\_d(array2, calc, newEl);

show(array2, calc);

system("pause");

break;

}

case 5: {

cout << "Массив с удаленными числами\n";

del\_d(array2, calc);

show(array2, calc);

system("pause");

break;

}

case 6: {

cout << "Максимальное четное число массива: ";

int temp = maxEvenNum(array2, calc);

if (temp == 1) {

cout << "Четных чисел в массиве нет\n";

}

else {

cout << temp << "\n";

}

system("pause");

break;

}

default:

break;

}

}

system("pause");

return 0;

}

void create(unsigned long\* arr, int n) { //заполнение массива

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

cin >> arr[i];

}

}

void show(unsigned long\* arr, int n) { //вывод массива

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

cout << arr[i] << " ";

}

cout << endl;

}