ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ УФИМСКИЙ УНИВЕРСИТЕТ НАУКИ И ТЕХНОЛОГИЙ

Кафедра ВМиК

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3**

по предмету **«Языки программирования»**

Выполнил: студент группы МО-203б

Ярошко Е. В.

Проверил:  
ст. преподаватель

Рипатти А. В.

**Уфа 2025 г.**

**Цель работы:**

Изучить работу с указателями в языке программирования С++.

**Задачи работы:**

- освоить понятие указателя

- освоить работу с указателями на примере линейного и двоичного поиска

**Задание:**

Оба задания делаются используя только арифметику указателей

1. Реализовать функцию с сигнатурой

int \* linear\_search( int \* begin, int \* end, int value );

где

begin – указатель на первый элемент массива

end – указатель на элемент массива после последнего

value – искомое значение

результат – указатель на первое вхождение элемента, равного value; если такого элемента нет – вернуть end

Поиск работает за линейное время, то есть нужный элемент ищется обычным проходом по массиву

2. Реализовать функцию двоичного поиска с сигнатурой

int \* binary\_search( int \* begin, int \* end, int value );

где

begin – указатель на первый элемент массива

end – указатель на элемент массива после последнего

value – искомое значение

результат – указатель на первое вхождение элемента, больше или равного value; если такого элемента нет – вернуть end

Предполагается, что массив, подаваемый на вход отсортирован (можно подавать на вход и не отсортированный массив, тогда результат вывода не до конца предсказуем, но программа не должна падать). Время работы O(log N), где N – длина массива.

3. Реализовать пользовательский интерфейс со следующим набором команд:

1. Создание массива. Пользователь вводит число N и программа генерирует массив длины N при помощи оператора new [], который затем заполняет случайными числами и запоминает указатель на массив и его длину. При повторном вызове данной команды старый массив должен удаляться при помощи оператора delete [].

2. Линейный поиск. Пользователь вводит искомый элемент и программа ищет его при помощи уже реализованной функции linear\_search.

3. Сортировка массива. Следует использовать сортировку из библиотеки <algorithm> с сигнатурой (если абстрагироваться от шаблонов и дополнительных аргументов) void sort( int \* begin, int \* end );

4. Перемешивание массива. Следует использовать функцию random\_shuffle, либо же реализовать самостоятельно (через указатели).

5. Двоичный поиск. Пользователь вводит искомый элемент x и программа ищет минимальный элемент, больший или равный x, при помощи уже реализованной функции binary\_search.

После каждой команды программа должна выводить текущий массив.

**Теоретические основы:**

В рамках данной лабораторной работы повсеместно используется понятие указателя.

Указатель — это переменная, хранящая в себе адрес на другую ячейку в памяти. Благодаря указателям можно ссылаться на данные в других ячейках, тем самым локализуя её значение динамически. Это позволяет создавать динамические объекты, управлять памятью вручную и создавать функции, способные изменять данные в передаваемых аргументах.

Чтобы использовать указатель на практике, необходимо освоить синтаксис инициализации такой переменной, операцию разыменования и присваивания значения.

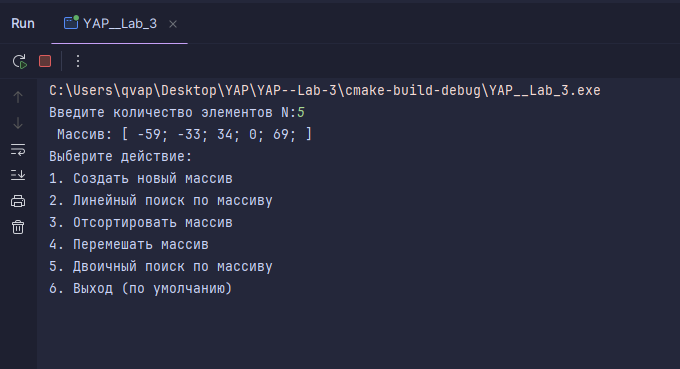
Чтобы указать, что переменная при инициализации является указателем, используется знак “\*”. Чтобы присвоить данной переменной адрес другой переменной, используется оператор “&”, означающий “получить адрес в памяти”.

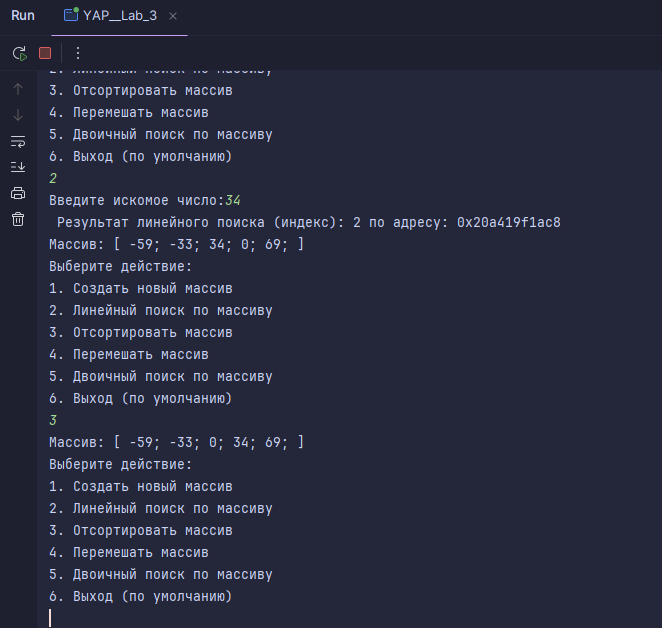
Для разыменования указателя и доступа к содержимому ячейки, на которую он указывает, используется оператор разыменовывания “\*”.

Списки в C++ хранятся в памяти в отдельных ячейках, и если в указатель передать ссылку на список, то в указателе сохранится адрес на первый элемент массива [0]. При добавлении единицы к указателю, адрес в нём сместится на одну ячейку, тем самым он станет указывать на следующий элемент массива [1]. Данная арифметика указателей позволяет нам проводить итерации со списком и, в контексте данной ЛР, проводить поиск элементов в списке и возвращать указатель на данный элемент.

**Листинг программы:**

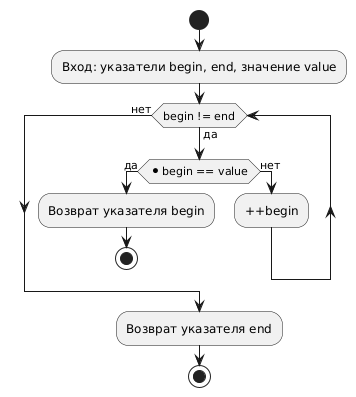
*#include* <iostream>  
*#include* <locale>  
*#include* <windows.h>  
*#include* <algorithm>  
*#include* <random>  
  
using namespace std;  
  
  
class *CSearch* {  
public:  
 static int\* *linear\_search* (int\* *begin*, int\* *end*, const int *value*) {  
 while (*begin* != *end*) {  
 if (\**begin* == *value*) {  
 return *begin*;  
 }  
 ++*begin*;  
 }  
 return *end*;  
 }  
  
 static int\* *binary\_search* (int\* *begin*, int\* *end*, const int *value*) {  
 while (*begin* != *end*) {  
 if (\**begin* >= *value*) {  
 return *begin*;  
 }  
 ++*begin*;  
 }  
 return *end*;  
 }  
};  
  
class *Interface* {  
private:  
 int N;  
 int\* arr;  
public:  
 *Interface*() {  
 N = 0;  
 arr = nullptr;  
 }  
 *~Interface*() {  
 delete[] arr;  
 }  
 void *display\_list*() {  
 if (this->arr == nullptr) {  
 cout *<<* "Массив пуст." *<< endl*;  
 return;  
 }  
 cout *<<* "Массив: [ ";  
 for (int i = 0; i < N; ++i) {  
 cout *<<* \*(arr + i) *<<* "; ";  
 }  
 cout *<<* "]" *<< endl*;  
 }  
  
 void *create\_list*() {  
 delete[] arr;  
  
 cout *<<* "Введите количество элементов N: ";  
 cin *>>* this->N;  
  
 this->arr = new int[N];  
  
 *// Заполнение случайными числами* for (int i = 0; i < N; ++i) {  
 \*(arr + i) = *rand*() % 200 - 100;  
 }  
  
 *menu*();  
 }  
  
 void *sort\_list*() {  
 *sort*(arr, arr + N);  
 *menu*();  
 }  
  
 void *shuffle\_list*() {  
 *shuffle*(arr, arr + N, *mt19937(random\_device()())*);  
 *menu*();  
 }  
  
 void *linear\_search*() {  
 int value = 0;  
 cout *<<* "Введите искомое число: ";  
 cin *>>* value;  
 int\* found = *CSearch*::*linear\_search*(arr, arr + N, value);  
 cout *<<* "Результат линейного поиска (индекс): " *<<* (found - arr) *<<* " по адресу: " *<<* found *<< endl*;  
 *menu*();  
 }  
  
 void *binary\_search*() {  
 int value = 0;  
 cout *<<* "Введите искомое число: ";  
 cin *>>* value;  
 const int\* found = *CSearch*::*binary\_search*(arr, arr + N, value);  
 cout *<<* "Результат двоичного поиска (индекс): " *<<* (found - arr) *<<* " по адресу: " *<<* found *<< endl*;  
 *menu*();  
 }  
  
 void *menu*() {  
 *display\_list*();  
 int choice = 0;  
 cout *<<* "Выберите действие: " *<< endl*;  
 cout *<<* "1. Создать новый массив" *<< endl*;  
 cout *<<* "2. Линейный поиск по массиву" *<< endl*;  
 cout *<<* "3. Отсортировать массив" *<< endl*;  
 cout *<<* "4. Перемешать массив" *<< endl*;  
 cout *<<* "5. Двоичный поиск по массиву" *<< endl*;  
 cout *<<* "6. Выход (по умолчанию)" *<< endl*;  
 cin *>>* choice;  
 switch (choice) {  
 case 1:  
 this->*create\_list*();  
 break;  
 case 2:  
 this->*linear\_search*();  
 break;  
 case 3:  
 this->*sort\_list*();  
 break;  
 case 4:  
 this->*shuffle\_list*();  
 break;  
 case 5:  
 this->*binary\_search*();  
 break;  
 default:  
 cout *<<* "Выход из программы..." *<< endl*;  
 break;  
 }  
 }  
};  
  
int *main*() {  
 *SetConsoleCP*(CP\_UTF8);  
 *SetConsoleOutputCP*(CP\_UTF8);  
 *setlocale*(LC\_ALL, ".UTF8");  
  
 auto\* ui = new *Interface()*;  
 ui->*create\_list*();  
  
 delete ui; *// Вызовется, когда ui выйдет из цикла* return 0;  
}

**Запуск программы:**

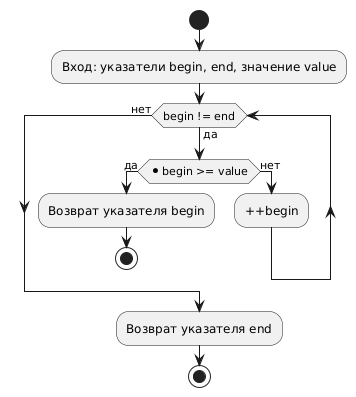


**Блок-схемы:**

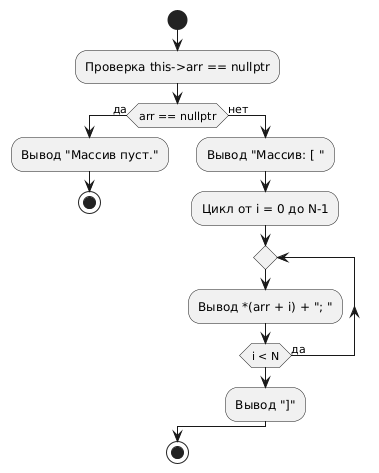
linear\_search:



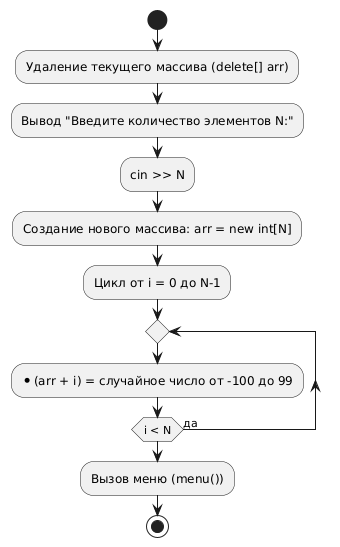
binary\_search:



display\_list:



create\_list:



menu:

