Практика 1.

Реализовать структуру данных «Динамический массив» и набор функций для работы с ней.

Динамический массив предназначен для хранения бесконечного (теоретически) количества однотипных данных. Необходимо обеспечить безопасность функций и всей программы в целом.

Необходимо реализовать следующие функции:

- Функция создания и инициализации полей массива (length, capacity, array)
- Добавления элемента в массив
- Удаление элемента из массива
- Вставка элемента в начало
- Вставка элемента в конец
- Вставка после определенного элемента
- Сортировка массива
- Линейный поиск элемента в массиве
- Бинарный поиск элемента в массиве

При работе с массивом предполагается, что изначально выделяется буффер размера по умолчанию (4 или 8). Затем работа с элементами массива идет через реализованные функции.

Программу необходимо оформить в виде меню. После запуска выводится список того, что можно сделать с массивом. Любой пункт можно выбирать множество раз.

Требования к коду:

Код должен быть написан согласно стандарту оформления кода RSDN (https://rsdn.org/article/mag/200401/codestyle.XML). Обращайте внимание на разделы кроме разделов с `class`.

Требования к среде разработки:

Предпочтительная IDE - Visual Studio. Можно любую другую, но в случае ошибок/вопросов помощи будет меньше.

Что необходимо повторить:

- Работу со структурами (struct)
- Создание функций, передача значений в функцию по значению/ссылке/указателю
- Простейшие сортировки (https://tproger.ru/translations/sorting-for-beginners/)
- Линейный поиск

Что необходимо изучить:

- Операторы new и delete
- Создание и работа с динамическими массивами на языке C++
- Бинарный поиск (https://tproger.ru/problems/searching-element-in-array/)

Работа со структурами.

Структура — это, некое объединение различных переменных (даже с разными типами данных), которому можно присвоить имя. Например, можно объединить данные об объекте Дом: город (в котором дом находится), улица, количество квартир, интернет (проведен или

нет) и т.д. в одной структуре. В общем, можно собрать в одну совокупность данные обо всем, что угодно, точнее обо всем, что необходимо конкретному программисту.

В языке C++ структуры создаются ключевым словом struct.

```
struct Person
{
	string name;
	string surename;
	int birthYear;
}
```

Обращение к элементам структур осуществляется помощью операторов «.» и «->». Если работа идет с объектом значением, то используется «.». Если работа идет с указателем на объект, то используется оператор «->».

```
void someFunc1()
{
	Person pers;
	pers.name = “Jonh”;
	pers.surename = “Smith”;
	pers.birthYear = 1990;
}

void someFunc1()
{
	Person* pers = new Person();
	pers->Name = “Jonh”;
	pers->Surename = “Smith”;
	pers->BirthYear = 1990;
	delete pers;
}
```

Передача объектов структур в функции можно осуществить 2 способами: по указателю и по ссылке. По сути в обоих случаях в функцию передается адрес переменной в памяти. Разница в том, что при передаче по указателю, в функции можно переназначить переменную, и потерять исходную переменную, в случае с передачей по ссылке это исключено. Помимо этого, есть разница в синтаксисе работы с переменной. При этом все же большие объекты (структуры с большим количеством полей) передают через указатели, это связано с тем, что в помышленном программировании такие объекты выделяются в динамической памяти, а для этого используются указатели. Поэтому, фактически на практике у нас почти всегда идет работа с указателями. Массивы передаются ТОЛЬКО по указателю.

```
void setBirthYear1(Person* pers, int year)
{
	Pers->birthYear = 1990;
}

void setBirthYear2(Person& pers, int year)
{
	Pers.birthYear = 1990;
```

```
}

void someFunc()
{
	Person p1; //переменная в стеке
	setBirthYear1(&p1);
	setBirthYear2(p1);

	Person* p2 = new Person(); //указатель на переменную в динамической памяти
	setBirthYear1(p2);
	setBirthYear2(&(*p2));
	delete p2;
}
```