Лабораторная работа № 10

Модуль работы с одномерными динамически создаваемыми массивами

Модуль на C++ логически состоит из двух файлов:

заголовочного файла (header file) с расширением *.h, который включает прототипы функций, объявление переменных и констант, о которых должны знать остальные модули, подключение необходимых для реализации заголовочных файлов;

файла с исходным кодом (source file) с расширением *.cpp, который содержит определение функций.

Упражнения

1. Создайте новый проект с именем ArrayFunctions в решении с именем Lab9.

2. В контекстном меню проекта выберите пункт Добавить, а затем – Создать элемент… или комбинацию клавиш Ctrl + Shift + A.

В диалоговом окне будут доступны три вида шаблонов файлов (рис. 1).

Для создания модуля работы с массивами нам понадобятся два из них: Файл C++ (.cpp) и Файл заголовка (.h).

3. Создайте файлы **array.cpp** и **array.h**.

Имя файлов модуля должно отражать его функциональные возможности. В нашем случае – это работа с массивами.

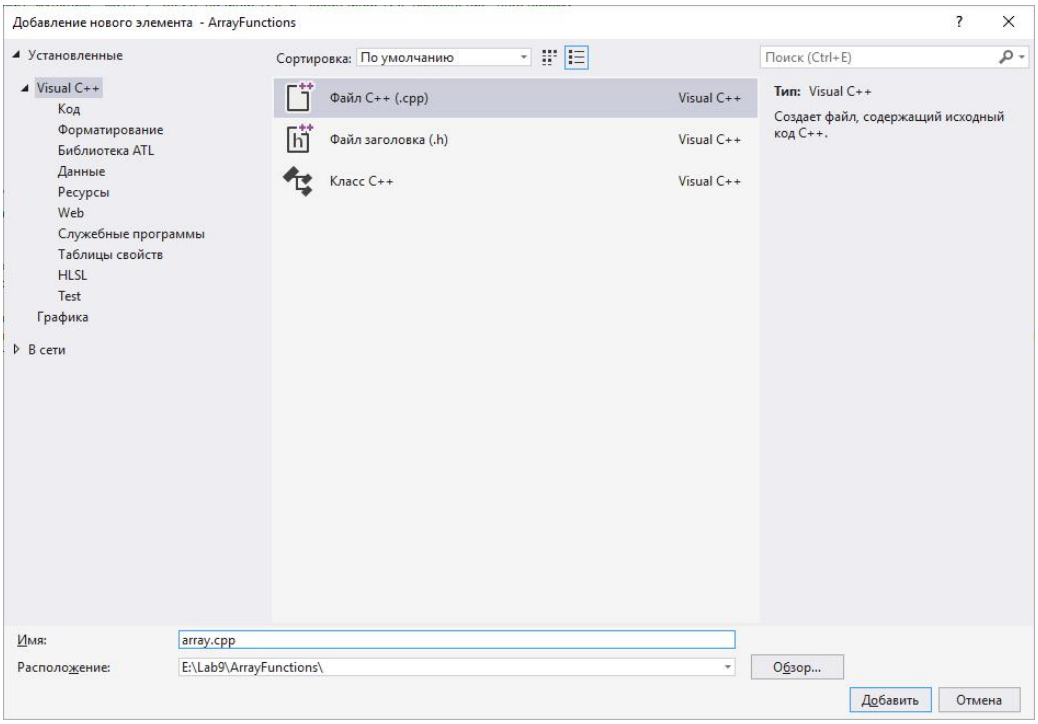

Рис. 1. Диалоговое окно добавления нового элемента

Созданные файлы должны занять место в соответствующих разделах проекта: array.h – в разделе заголовочных файлов, array.cpp – среди файлов исходного кода (рис. 2).

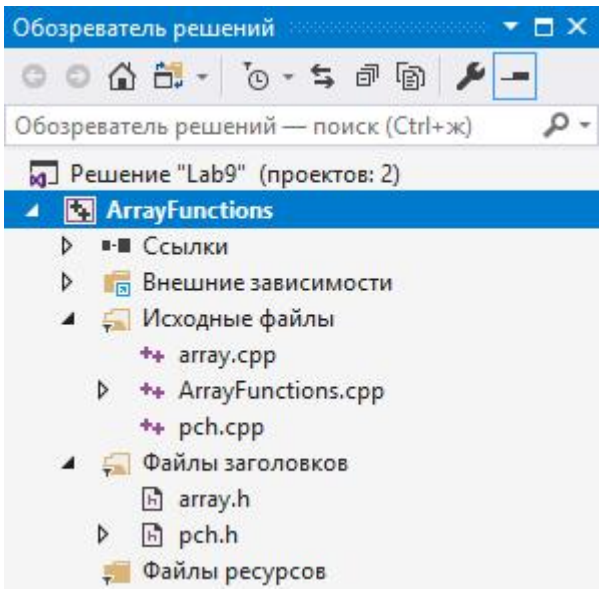

Рис. 2. Окно обозревателя решений

Модуль должен подключаться к программе один раз с помощью директивы компилятора **#include**, после которой указывается имя заголовочного файла этого модуля. Во избежание проблем с «двойным подключением» применяется один из следующих механизмов:

директива препроцессора **#pragma once**, которая по умолчанию уже присутствует в заголовочном файле при его создании в среде программирования MS Visual Studio (см. содержимое файла array.h);

особая синтаксическая конструкция, называемая #include guards (защита подключения), иногда также называемая macro guard или макрозащита (см. содержимое файла pch.h).

Для данного упражнения эта конструкция может иметь следующий вид:

```
#ifndef ARRAY_H
#define ARRAY_H
...
#endif //ARRAY_H
```

Для использования создаваемого модуля в файле ArrayFunctions.cpp должна содержаться директива подключения заголовочного файла

```
#include "array.h"
```

4. В файл ArrayFunctions.cpp добавьте директиву подключения array.h.

При подключении этого заголовочного файла компилятор проверяет, был ли ранее определён идентификатор **ARRAY_H**. При первом подключении макропеременная **ARRAY_H** определяется с помощью директивы **#define**, после чего выполняется основная часть заголовочного файла.

Если же этот заголовочный файл уже был подключен ранее, то **ARRAY_H** уже была определена. В таком случае, содержимое файла **array.h** будет проигнорировано.

5. Включите в заголовочный файл объявления функций работы с массивом:

```
void initConsol(int *, int); //формирование из консоли
void initRandom(int *, int, int, int);
//заполнение случайными числами
void printConsol(int *, int); //вывод в консоль
```

6. В файл *Array.cpp* внесите определения функций работы с массивом:

```
void initConsol(int *array, int n)
{
	for (int i = 0; i < n; i++) {
		cout << "Input " << i << " element: ";
		cin >> *(array+i);
	}
}

void initRandom(int *array, int n, int a, int b)
```

```
{
	srand(time(0));
	for (int i = 0; i < n; i++) {
		*(array+i) = rand() % (b-a) +a;
	}
}

void printConsol(int *array, int n)
{
	for (int i = 0; i < n; i++) {
		cout << *(array+i) << " ";
	}
	cout << "\n";
}
```

7. В файле ArrayFunctions.cpp дополните функцию main():

```
#include "array.h"

int main()
{
	const int n = 10;
	int * a = new int [10];
	initConsol(a, n);
	printConsol(a, n);
	system("pause");
	return 0
	;
}
```

8. Выполните компиляцию проекта и протестируйте его работу.

Задание на лабораторную работу:

Разработать модуль работы с одномерными массивами, реализующий перечисленные функциональные возможности. В задачах, использующих условие, под условием понимается функция типа **`bool`**, принимающая на вход один элемент массива. Например, кратный трем, положительный, с суммой цифр не меньше 5 и т.п. Условие в функцию передается как указатель на функцию.

1. Чтение элементов массива с клавиатуры.
2. Заполнение массива с помощью датчика случайных чисел. Диапазон значений задает пользователь с клавиатуры.
3. Вывод элементов массива на экран.
4. Поиск заданного элемента. Функция должна возвращать номер элемента или –1, если он не найден.

5. Поиск максимального/минимального элемента в массиве. Если таких элементов несколько, то функция должна возвращать номер первого.

6. Поиск минимального/максимального элемента в массиве, удовлетворяющего заданному условию.

7. Поиск всех вхождений заданного элемента. Функция должна возвращать массив номеров или `nullptr`.

8. Сформировать массив из элементов исходного, удовлетворяющих заданному условию. Условие в функцию передается как указатель на функцию.

9. Удаление элемента с заданным индексом.

10. Удаление всех элементов, равных заданному.

11. Вставка нового элемента на заданное место.

12. Удаление из массива *K* элементов, начиная с заданного индекса. Выполнить проверку введенного индекса и числа *K* на корректность.

13. Удаление элементов массива, удовлетворяющих заданному условию.

14. Вставка в массив *K* элементов, начиная с заданного номера. Новые элементы вводятся с клавиатуры.