Занятие № 3.

Функции C++

1. Объявление функций

Функция – это отдельный блок кода, который выполняется при вызове из других функций. Функции могут принимать на вход любое количество аргументов или не иметь аргументов. Также функция может возвращать одно значение или не возвращать значения.

Рассмотрим пример простейшей функции, которая не принимает аргументов и не возвращает значений.

```
#include <iostream>

using namespace std;

void hello(); // определяем функцию

int main()
{
     cout << "\nHello, World!\n";

     hello(); // вызываем функцию

     return 0;
}

// реализуем функцию
void hello()
{
    cout << "\nHello!\n";
```

```
}
```

Рассмотрим теперь функцию, которая принимает аргументы и возвращает результат.

```
#include <iostream>

using namespace std;

double power(double x, int n); // определяем функцию

int main()
{
    setlocale(LC_ALL, "Russian");// устанавливаем русскую кодировку

    double x;
    int n;

     cout << "\nВведите число > ";
     cin >> x;

     cout << "\nВведите степень > ";
     cin >> n;

   double xn = power(x, n); // вызываем функцию и получаем результат

   cout << "\nЧисло " << x << " в степени " << n << " равно " << xn << "\n\n";

     return 0;
}

// реализуем функцию
double power(double x, int n)
{
```

```
    double res = 1; // определяем переменную, которая будет
результатом

    for (int i = 0; i < n; i++)
    {
        res = res * x; // вычисляем степень
    }

    return res; // возвращаем результат
}
```

2. Аргументы функции

Передаваемые в функцию аргументы не могут быть изменены в результате работы функции, хотя они могут быть изменены внутри самой функции.

```
#include <iostream>

using namespace std;

void func(int x); // определяем функцию

int main()
{
    setlocale(LC_ALL, "Russian");//устанавливаем русскую кодировку

    double x = 5; // определяем и задаем значение переменной

    cout << "До вызова функции x = " << x;

    func(x);// Вызываем функцию, в которой меняем значение аргумента

    cout << "\n\nПосле вызова функции x = " << x;
```

```
    cout << "\n\n";

     return 0;
}

// реализуем функцию
void func(int x)
{
    x = 3; // Меняем значение аргумента

    cout << "\n\nПосле изменения значения внутри функции x = " << x;
}
```

3. Область видимости переменных

Переменные бывают локальными и глобальными. Локальные переменные объявляются внутри функции или блока { }. И имеют область видимости, т.е. могут быть использованы только внутри функции или блока, в котором они объявлены.

Глобальные переменные объявляются вне функций и могут быть доступными во всех функциях.

```
#include <iostream>

using namespace std;

void func(); // определяем функцию

int a = 5; // создаем глобальную переменную

int main()
{
```

```
    setlocale(LC_ALL, "Russian");// устанавливаем русскую кодировку

    cout << "Значение a = " << a; // глобальную переменную можно использовать

     a = 6; // и менять

     cout << "\n\nЗначение после изменения a = " << a;

     func(); // вызываем функцию

     cout << "\n\nЗначение после изменения в функции a = " << a;

     cout << "\n\n";
     return 0;
}

// реализуем функцию
void func()
{
     a++; // Меняем значение глобальной переменной
}
```

4. Сокрытие переменных

В случае, когда имя локальной переменной совпадает с именем глобальной переменной, глобальная переменная будет не доступна в области видимости локальной переменной – это называется сокрытие переменной.

```
#include <iostream>

using namespace std;

void func(); // определяем функцию
```

```
int a = 5; // создаем глобальную переменную

int main()
{
    setlocale(LC_ALL, "Russian"); //устанавливаем русскую кодировку

    int a; // создаем локальную переменную

    a = -10; // прискаеваем значение локальной переменной

    cout << "\nB main: Значение a = " << a;

    func(); // вызываем функцию

    cout << "\nB main: Значение a = " << a;

    cout << "\n\n";

    return 0;
}

// реализуем функцию
void func()
{
    cout << "\nB func: Значение переменной a = " << a;

    cout << "\nB func: Увеличиваем значение переменной";

    a++; // меняем значение глобальной переменной
}
```

5. Разделение программы на файлы

Большие проекты обычно содержат сотни тысяч строк кода, поэтому при создании больших программ необходимо использовать разделение кода программы на несколько файлов.

Для разделения программы на несколько файлов, необходимо объединить их в один проект. Покажем, как это делается в Dev-C++.

1. Выбираем в меню: Файл → Создать → Проект…

Откроется форма создания проекта:

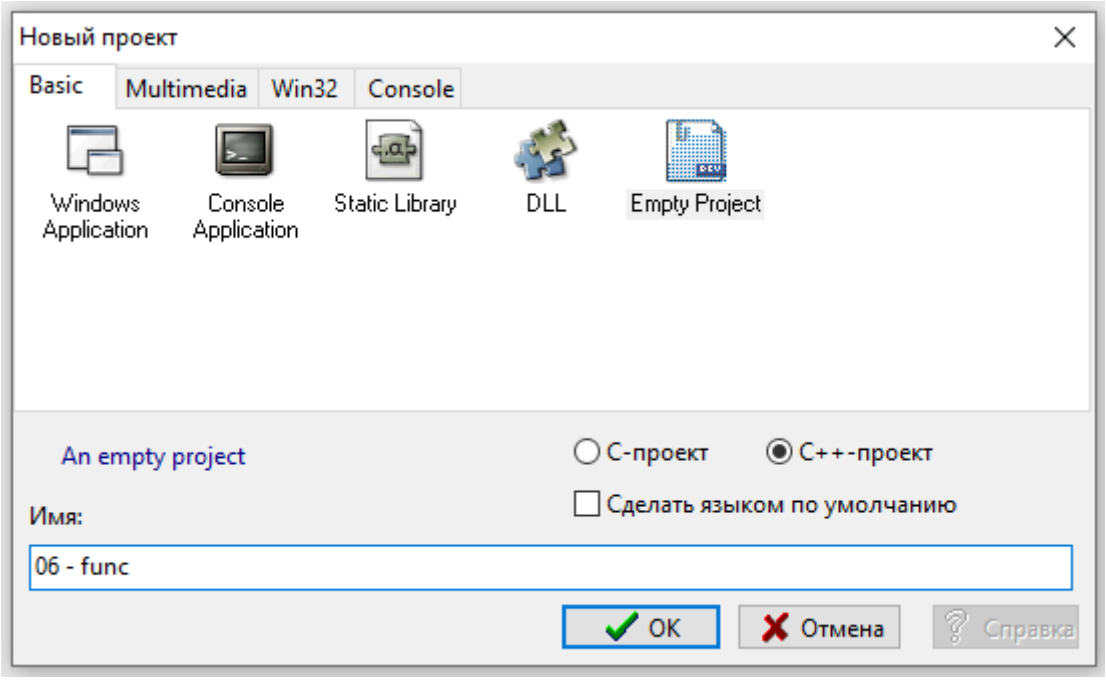

Рис. 1. Форма создания проекта.

Выбираем “Empty Project”, C++ - проект, и вводим имя проекта.

2. Выбираем папку, в которой будет проект.

3. В меню: Проект → Добавить к проекту

Создаем файл main.cpp – это будет наш головной файл.

4. Добавляем в проект еще два файла: func.h и func.cpp.

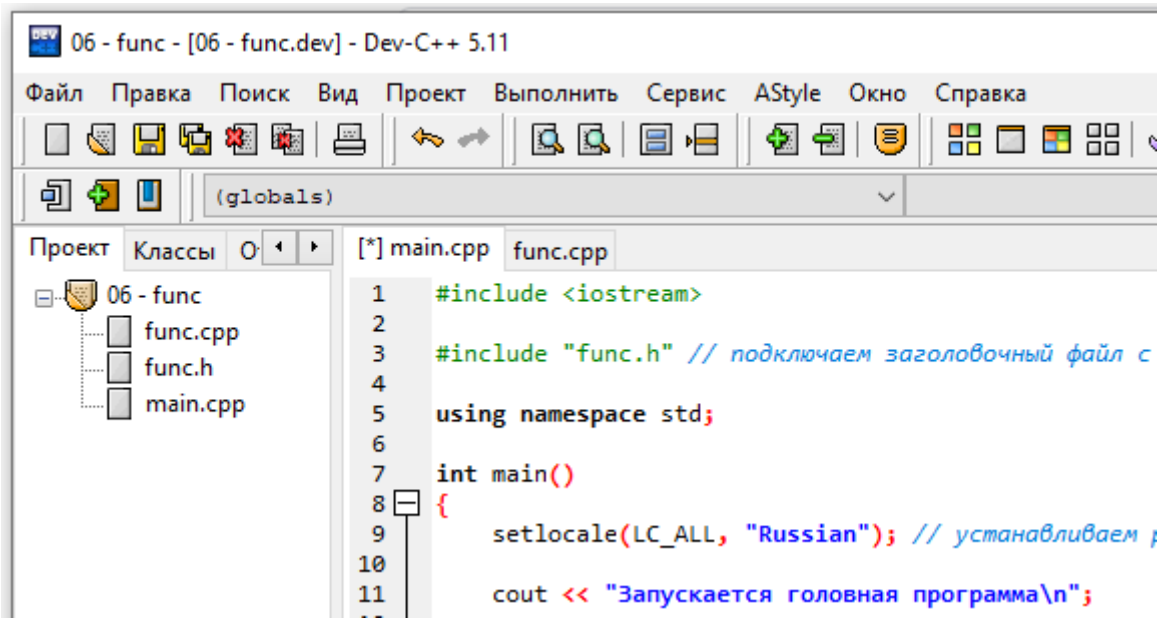

Рис. 2. Файлы проекта.

Переход между файлами проекта можно сделать в левой части среды Dev-C++, во вкладке «Проект».

Заполним файлы:

main.cpp

```
#include <iostream>

#include "func.h" // подключаем заголовочный файл с внешней
//функцией

using namespace std;

int main()
{
	setlocale(LC_ALL, "Russian"); // устанавливаем русскую кодировку
```

```
    cout << "Запускается головная программа\n\n";

    cout << "Введите число > ";

    int n;

    cin >> n;

    int m;

    m = func(n); // вызов функции из другого файла

    cout << "\nРезультат работы внешней функции: " << m << "\n\n";

   return 0;
}
```

func.h

```
int func(int n); // объявляем функцию
```

func.cpp

```
// реализация функции
int func(int n)
{
    int res = 0;

    for(int i = 1; i <= n; i++)
    {
        res += i; // операция += эквивалентна res = res + i;
    }
```

```
    return res;
}
```

Компилируем и запускаем проект!