Лабораторная работа №4. Функции

Простые функции

Наш первый пример демонстрирует простую функцию, которая печатает строку из 45 символов *. В программе создается таблица, и для того, чтобы сделать эту таблицу удобочитаемой, используются разделяющие линии, состоящие из символов *.

Table 1.

```
// table.cpp
// demonstrates simple function
#include <iostream>
using namespace std;
//-------------------------------------------------------
// определение функции starline()
void starline()
// заголовок функции
{
	for (int j = 0; j < 45; j++) // тело функции
		cout << '*';
	cout << endl;
}
int main()
{
	setlocale(LC_ALL, "Russian");
	starline(); // вызов функции
	cout << "Тип данных Диапазон" << endl;
	starline(); // вызов функции
	cout << "char -128...127" << endl
		<< "short - 32 768...32 767"  << endl
		<<"int Системно - зависимый"<< endl
		<<"long -2 147 483 648...2 147 483 647" << endl;
	starline();// вызов функции
	return 0;
}
```

Table 2.

```
// table.cpp
// demonstrates simple function
#include <iostream>
using namespace std;
//-------------------------------------------------------
// определение функции starline()
void starline(); // объявление функции (прототип)
int main()
{
	setlocale(LC_ALL, "Russian");
	starline(); // вызов функции
	cout << "Тип данных Диапазон" << endl;
	starline(); // вызов функции
	cout << "char -128...127" << endl
		<< "short - 32 768...32 767"  << endl
		<<"int Системно - зависимый"<< endl
		<<"long -2 147 483 648...2 147 483 647" << endl;
	starline(); // вызов функции
	return 0;
}
void starline()
{
	for (int j = 0; j < 45; j++) // тело функции
		cout << '*';
	cout << endl;
}
```

Определение функции состоит из заголовка и тела функции. Тело функции состоит из последовательности операторов, заключенной в фигурные скобки. Заголовок функции должен соответствовать ее прототипу: **имя функции и тип возвращаемого ей значения должны совпадать с указанными в прототипе**; кроме того, **аргументы** функции, если они есть, **должны иметь те же типы и следовать в том же порядке, в каком они указывались в прототипе.**

Передача констант в функцию

В качестве примера рассмотрим следующую ситуацию. Пусть нам необходимо сделать функцию starline() более гибкой: мы хотим изменить ее таким образом, чтобы она выводила не фиксированное (равное 45 в предыдущих примерах), а любое заданное количество символов. В следующем примере, TABLEARG, мы создаем модификацию функции starline() под названием repchar(). В качестве аргументов для функции используется символ, предназначенный для печати, а также число раз, которое данный символ будет выведен на экран.

```
// table.cpp
// demonstrates simple function
#include <iostream>
using namespace std;
//-----------------------------------------------------
// определение функции starline()
void repchar(char, int);   // объявление функции (прототип)
int main()
{
	setlocale(LC_ALL, "Russian");
	repchar('-', 43);  // вызов функции
	cout << "Тип данных Диапазон" << endl;
	repchar('=', 23);  // вызов функции
	cout << "char -128...127" << endl
		<< "short - 32 768...32 767"  << endl
		<<"int Системно - зависимый"<< endl
		<<"long -2 147 483 648...2 147 483 647" << endl;
	repchar('-', 43);   // вызов функции
	return 0;
}
void repchar(char ch, int n)
{
	for (int j = 0; j < n; j++) // тело функции
		cout << ch;
	cout << endl;
}
```

Передача значений переменных в функцию

В примере TABLEARG роль аргументов выполняли константы: '-', 43 и т. д. Теперь давайте рассмотрим пример, в котором вместо констант в функцию будут передаваться значения переменных. Следующая программа, VARARG, включает в себя ту же функцию repchar(), что и программа TABLEARG, но теперь символ и число его повторений будут задаваться пользователем:

```
// vararg.cpp
// передача переменных в функцию в качестве аргументов
#include <iostream>
using namespace std;
void repchar(char, int);
int main()
{
	setlocale(LC_ALL, "Russian");
	char chin;
	int nin;
	cout << "Введите символ: ";
	cin >> chin; // объявление функции
	cout << "Введите число повторений символа: ";
	cin >> nin;
	repchar(chin, nin);
	return 0;
}
// определение функции repchar()
void repchar(char ch, int n)
{

	for (int j = 0; j < n; j++) // тело функции
		cout << ch;
	cout << endl;
}
```

Структурные переменные в качестве аргументов

Структура Distance

Следующий пример, ENGLDISP, содержит функцию, которая использует аргумент типа Distance — структуры, фигурировавшей в нескольких примерах из главы 4 «Структуры».

```
// engldisp.cpp
// передача структурных переменных в функцию
#include <iostream>
using namespace std;
////////////////////////////////////////////////////////
struct Distance // длина в английской системе
{
	int feet;
	float inches;
};
////////////////////////////////////////////////////////
void engldisp(Distance);  // объявление
int main()
{
	setlocale(LC_ALL, "Russian");
	Distance d1, d2; // определение двух длин
	cout << "Введите число футов: "; cin >> d1.feet; // ввод значений полей d1
	cout << "Введите число дюймов: "; cin >> d1.inches;
	cout << "\nВведите число футов: "; cin >> d2.feet; // ввод значений полей d2
	cout << "Введите число дюймов: ";  cin >> d2.inches;
	cout << "\nd1 = ";
	engldisp(d1); // вывод значения d1
	cout << "\nd2 = ";
	engldisp(d2); // вывод значения d2
	cout << endl;
	return 0;
}
//-----------------------------------------
// функция engldisp(), отображающая значения
// полей структурной переменной типа Distance
void engldisp(Distance dd) // параметр dd типа Distance
{
	cout << dd.feet << "\'-" << dd.inches << "\"";
}
```

Значение, возвращаемое функцией

Когда выполнение функции завершается, она может возвратить значение программе, которая ее вызвала. Как правило, возвращаемое функцией значение имеет отношение к решению задачи, возложенной на эту функцию. Следующий пример демонстрирует применение функции, получающей значение веса в фунтах и возвращающей эквивалентное значение этого веса в килограммах:

```
// convert.cpp
// демонстрирует механизм возврата значения функцией
#include <iostream>
using namespace std;
float lbstokg(float);
// прототип функции
int main()
{
	setlocale(LC_ALL, "Russian");
	float lbs, kgs;
	cout << "\nВведите вес в фунтах: ";
	cin >> lbs;
	kgs = lbstokg(lbs);
	cout << "Вес в килограммах равен " << kgs << endl;
	return 0;
}
// функция lbstokg()
// переводит фунты в килограммы
float lbstokg(float pounds)
{
	float kilograms = 0.453592 * pounds;
	return kilograms;
}
```

Задачи

1. Написать функцию, которая вычисляет объем цилиндра. Параметрами функции должны быть радиус и высота цилиндра.
2. Написать функцию, которая возвращает максимальное из двух целых чисел, полученных в качестве аргумента.
3. Написать функцию, которая сравнивает два целых числа и возвращает результат сравнения в виде одного из знаков: >, < или =.
4. Написать функцию Procent, которая возвращает процент от полученного в качестве аргумента числа.
5. Напишите функцию с именем circarea(), которая вычисляет площадь круга. Функция должна принимать один аргумент типа float и возвращать значение типа float. Напишите функцию main(), которая просит пользователя ввести значение радиуса, вызывает функцию circarea(), а затем отображает результат вычисления на экране.
6. Возведение числа n в степень p — это умножение числа n на себя p раз. Напишите функцию с именем power(), которая в качестве аргументов принимает значение типа double для n и значение типа int для p и возвращает значение типа double. Для аргумента, соответствующего степени числа, задайте значение по умолчанию, равное 2, чтобы при отсутствии показателя степени при вызове функции число n возводилось в квадрат. Напишите функцию main(), которая запрашивает у пользователя ввод аргументов для функции power(), и отобразите на экране результаты ее работы.
7. Напишите функцию с именем hms_to_secs(), имеющую три аргумента типа int: часы, минуты и секунды. Функция должна возвращать эквивалент переданного ей временного значения в секундах (типа long). Создайте программу, которая будет циклически запрашивать у пользователя ввод значения часов, минут и секунд и выводить результат работы функции на экран.