ФУНКЦИИ

Цель работы: научиться задавать свои функции и изучить правила работы с ними.

Теоретические сведения

В ранее рассмотренных примерах неоднократно использовались различные функции подключаемых библиотек. Вместе с тем существующих функций языка С недостаточно для написания собственных программ и возникает необходимость создания своих функций. В связи с этим нужно понимать, в каких случаях целесообразно создавать свои функции. Обычно это делается для избавления много раз писать один и тот же код в программе. Например, если часто выполняются действия копирования одной строки в другую, то такую операцию лучше определить в виде функции и использовать ее по мере необходимости. Для объявления функции используется следующий синтаксис:

```
<тип> <имя функции> ([список параметров]) { <тело функции> }
```

Тип определяет возвращаемый тип функции. Имя функции служит для ее вызова в программе и ее правило определения совпадает с правилом определения имен переменных. Список параметров необходим для передачи функции каких-либо данных при ее вызове. Телов функции – это набор операторов, которые выполняются при ее вызове. Следующий пример показывает правило определения пользовательских функций.

Листинг 7. Пример задания функции.

```
double square(double x)
{
 x = x*x;
 return x;
}
int main()
{
 double arg = 5;
 double sq1=square(arg);
 double sq2=square(3);
 return 0;
}
```

В данном примере задается функция с именем square, которая принимает один входной параметр типа double, возводит его в квадрат и возвращает вычисленное значение вызывающей программе с помощью оператора return. Следует отметить, что работа функции завершается при вызове оператора

return. Даже если после этого оператора будут находиться другие операторы, то они выполняться не будут. Например,

```
int square(int x)
{
 x = x*x;
 return x;
 printf("%d",x);
}
```

при вызове данной функции оператор printf() не будет выполнен никогда, т.к. оператор return завершит работу функции square. Оператор return является обязательным, если функция возвращает какие-либо значения. Если же она имеет тип void, т.е. ничего не возвращает, то оператор return может не использоваться.

Пользуясь рассмотренными правилами, можно создавать множество своих функций. При этом важно, чтобы объявление функции было раньше ее использования в программе подобно переменным. Именно поэтому во всех примерах объявление функций осуществляется до функции main(), в которой они вызываются.

Функция может принимать произвольное число аргументов, но возвращает только один или не одного (тип void). Для задания нескольких аргументов функции используется следующая конструкция:

```
void show(int x,int y,int z) {}
```

Здесь следует обратить внимание на то, что каждой переменной в списке аргументов функции предшествует ее тип. В отличие от объявления обычных переменных. Поэтому следующая программная строка приведет к сообщению об ошибке на этапе компиляции:

```
void show(int x, y, z) {}      //неверное объявление
```

Если число пользовательских функций велико (50 и выше), то возникает неудобство в их визуальном представлении в общем тексте программы. Действительно, имея список из 100 разных функций с их реализациями, в них становится сложно ориентироваться и вносить необходимые изменения. Для решения данной проблемы в языке С при создании своих функций можно пользоваться правилом: сначала задаются объявления функции, а затем их реализации.

Язык С позволяет задавать функции с одинаковыми именами, но разными типами входных аргументов. Следующий пример демонстрирует удобство использования таких функций при их вызове.

Листинг 8. Пример использования перегруженных функций.

```
#include <stdio.h>
double abs(double arg);
float abs(float arg);
int abs(int arg);
int main()
{
 double a_d = -5.6;
 float a_f = -3.2;
 int a_i;
 a_d = abs(a_d);
 a_f = abs(a_f);
 a_i = abs(-8);
 return 0;
}
double abs(double arg)
{
 if(arg < 0) arg = arg*(-1);
 return arg;
}
float abs(float arg)
{
 return (arg < 0) ? -arg : arg;
}
int abs(int arg)
{
 return (arg < 0) ? -arg : arg;
}
```

В представленной программе задаются три функции с именем abs и разными входными и выходными аргументами для вычисления модуля числа. Благодаря такому объявлению при вычислении модуля разных типов переменных в функции main() используется вызов функции с одним и тем же именем abs. При этом компилятор в зависимости от типа переменной автоматически выберет нужную функцию. Такой подход к объявлению функций называется перегрузкой.

В языке С можно задавать значения аргументов функции, которые будут использоваться по умолчанию, т.е. если программист не введет свое значение. Приведенный ниже фрагмент программы демонстрирует правило использования аргументов по умолчанию.

```
void some_func(int a = 1, int b = 2, int c = 3)
{
 printf("a = %d, b = %d, c = %d\n",a,b,c);
}
```

Благодаря начальной инициализации значений переменных, функция some_func() может быть вызвана с разным набором аргументов:

```
int main(void)
{
```

```
 show_func();
 show_func(10);
 show_func(10,20);
 show_func(10,20,30);
 return 0;
}
```

В результате, на экране появятся следующие строки:

a = 1, b = 2, c = 3
a = 10, b = 2, c = 3
a = 10, b = 20, c = 3
a = 10, b = 20, c = 30

Из полученного результата видно, что по умолчанию значения аргументов равны установленным значениям при определении функции. В случае ввода новых значений, переменные a, b и c соответственно меняют свои значения на введенные.

При использовании значений аргументов по умолчанию следует пользоваться правилом: аргументы со значениями по умолчанию должны находиться в списке аргументов функции последними. Следующий пример показывает правильные и неправильные объявления функций:

```
void my_func(int a, int b = 1, int c = 1);//правильное объявление
void my_func(int a, int b, int c = 1);     //правильное объявление
void my_func(int a=1, int b, int c = 1); //неправильное объявление
void my_func(int a, int b = 1, int c);    //неправильное объявление
```

В языке С допускается чтобы функция вызывала саму себя. Этот процесс называется рекурсией. В некоторых задачах программирования такой подход позволяет заметно упростить создаваемый программный код. Рассмотрим данный процесс на следующем примере.

Листинг 9. Пример использования рекурсивных функций.

```
#include <stdio.h>
void up_and_down(int );
int main(void)
{
 up_and_down(1);
 return 0;
}
void up_and_down(int n)
{
 printf("Уровень вниз %d\n",n);
 if(n < 4) up_and_down(n+1);
 printf("Уровень вверх %d\n",n);
}
```

Результатом работы этой программы будет вывод на экран следующих строк:

Уровень вниз 1
Уровень вниз 2
Уровень вниз 3
Уровень вниз 4
Уровень вверх 4
Уровень вверх 3
Уровень вверх 2
Уровень вверх 1

Полученный результат работы программы объясняется следующим образом. Вначале функция main() вызывает функцию up_and_down() с аргументом 1. В результате аргумент n данной функции принимает значение 1 и функция printf() печатает первую строку. Затем выполняется проверка и если n < 4, то снова вызывается функция up_and_down() с аргументом на 1 больше n+1. В результате вновь вызванная функция печатает вторую строку. Данный процесс продолжается до тех пор, пока значение аргумента не станет равным 4. В этом случае оператор if не сработает и вызовется функция printf(), которая печатает пятую строку «Уровень вверх 4». Затем функция завершает свою работу и управление передается функции, которая вызывала данную функцию. Это функция up_and_down() с аргументом n=3, которая также продолжает свою работу и переходит к оператору printf(), который печатает 6 строку «Уровень вверх 3». Этот процесс продолжается до тех пор, пока не будет достигнут исходный уровень, т.е. первый вызов функции up_and_down() и управление вновь будет передано функции main(), которая завершит работу программы.

Задание на лабораторную работу

1. Написать две программы по работе с функциями в соответствии с номером своего варианта.

Вариант	Задание 1	Задание 2
1	1.Написать функцию вычисления площади прямоугольника 2. Выделить данную функцию в заголовочный файл	Написать функцию вычисления *F(N) = F(N-1)*1.7/3.1,* с использованием рекурсии, при условии F(1)=1.
2	1.Написать функцию вычисления периметра прямоугольника 2. Выделить данную функцию в заголовочный	Написать рекурсивную функцию вычисления x^n

	файл	
3	1.Написать функцию вычисления длины окружности 2. Выделить данную функцию в заголовочный файл	Написать функцию вычисления *F(N) = F(N-1)*1.7/3.1,* с использованием рекурсии, при условии F(1)=1.
4	1.Написать функцию вычисления площади круга 2. Выделить данную функцию в заголовочный файл	Написать рекурсивную функцию вычисления x^n
5	1.Написать функцию вычисления объема параллелепипеда 2. Выделить данную функцию в заголовочный файл	Написать функцию вычисления *F(N) = F(N-2) + 3.5,* с использованием рекурсии, при условии F(1)=1, F(0)=0 .
6	1.Написать функцию вычисления евклидового расстояния между двумя точками 2. Выделить данную функцию в заголовочный файл	Напишите рекурсивную функцию f(n), принимая во внимание следующее: $\begin{cases} 6 & \text{if } n = 1, \\ \frac{1}{2} f_{n-1} + 4 & \text{if } n > 1. \end{cases}$
7	1.Написать функцию вычисления суммы элементов массива 2. Выделить данную функцию в заголовочный файл	Написать функцию вычисления *F(N) = F(N-2) + 3.5,* с использованием рекурсии, при условии F(1)=1, F(0)=0 .
8	1.Написать функцию нахождения максимального значения элемента массива 2. Выделить данную функцию в заголовочный файл	Написать функцию вычисления *F(N) = F(N-2) + 3.5,* с использованием рекурсии, при условии F(1)=1, F(0)=0 .
9	1.Написать функцию нахождения минимального значения элемента массива 2. Выделить данную функцию в заголовочный файл	Напишите рекурсивную функцию f(n), принимая во внимание следующее: $\begin{cases} 6 & \text{if } n = 1, \\ \frac{1}{2} f_{n-1} + 4 & \text{if } n > 1. \end{cases}$
10	1.Написать функцию вычисления произведения элементов массива 2. Выделить данную функцию в заголовочный файл	Написать функцию вычисления *F(N) = F(N-2) + 3.5,* с использованием рекурсии, при условии F(1)=1, F(0)=0 .

Содержание отчета

1. Титульный лист с названием лабораторной работы, номером варианта, фамилией студента и группы.
2. Текст программ.
3. Результаты действия программ.
4. Выводы о полученных результатах работы программ.

Контрольные вопросы

1. Запишите прототип функции, которая принимает два целочисленных аргумента и возвращает вещественное число.
2. Допустим, даны три функции:
   ```
   int abs(int x);
   float abs(float x);
   long abs(long x);
   ```
   Какая из этих трех функций будет вызвана в строке float a = abs(-6);?
3. Запишите функцию возведения числа в квадрат.
4. Дайте понятие рекурсии.
5. В каких задачах целесообразно использовать рекурсивные функции?
6. Приведите функцию с тремя аргументами, один из которых задан со значением по умолчанию.