Рязанский станкостроительный колледж РГРТУ

Основы алгоритмизации и программирования

Тема 5. Модульное программирование. Функции.

Рязань 2020

Оглавление

Функции

Общие сведения

Наряду со стандартными функциями пользователь может организовать свои собственные функции, которые в общем смысле можно называть подпрограммами.

Подпрограмма – это именованная часть программы, которая вызывается, получает параметры (аргументы), выполняет определённые действия и возвращает управление в точку вызова, возвращая полученный результат.

Вызов подпрограммы (или обращение к подпрограмме, или активизация подпрограммы) – это скачок по программному коду: если в тексте программы встречается обращение к подпрограмме, то выполнение текущей программы приостанавливается, осуществляется переход на код подпрограммы, выполняются её операторы, а затем осуществляется возврат в текущую программу в точку вызова.

Параметры подпрограммы – это величины, имеющие смысл входных и выходных данных, т.е. это исходные данные, получаемые подпрограммой, и результаты вычислений, возвращаемые в вызвавшую программу.

Подпрограммы описываются до описания главной функции или описываются в отдельном классе. Подпрограмма, используемая другой подпрограммой, должна описываться ранее (до того, как она будет использована в другой подпрограмме).

Структура подпрограммы

Структура любой подпрограммы (функции) аналогична структуре программы т.е. ее главной функции. Она имеет: 1) заголовок; 2) Описание переменных и операторы. Обратим внимание на тот факт, что внутри подпрограммы могут размещаться другие подпрограммы. Уровень вложения синтаксически не ограничен.

Синтаксическая форма функции имеет следующий вид:

```
[модификаторы] тип_результата имя_функции([список_параметров])
{
      объявления переменных;
      операторы;
      return результат;
}
```

где:

- ***Модификаторы*** — модификатор доступа к функции. В данном случае нужно указать ***static***, чтобы можно было использовать функцию сразу без создания объекта этого класса.
- ***Тип_результата*** — (int, double, void) тип результата ответа возвращаемого при использовании функции.
- ***Список_параметров*** – Имена входных и выходных переменных (аргументов), используемых в функции.
- ***Объявления переменных*** – Имена локальных переменных, используемых в функции.
- ***Операторы*** — некоторые операторы и выражения, содержащиеся внутри функции и выполняющиеся каждый раз при вызове функции. Внутри операторов мы можем обращаться к локальным переменным, объявленным внутри функции; а также к аргументам функции.
- ***return результат*** — оператор, останавливающий работу функции и возвращающий указанный результат в качестве её значения (при этом ***тип***

результата должен соответствовать типу результата в объявлении функции). Наличие этого оператора обязательно для функции, возвращающей значение. Для функции, объявленной как ***void*** этот оператор необязателен, но можно вызывать оператор ***return*** без аргументов, это досрочно завершит функцию, иначе — будут выполнены все действия до конца блока описания функции.

Пример 1.

Формулировка задачи:

Создадим и используем функции для сложения 2 чисел.

Методика решения задачи:

В данном случае т.к. результат функции один, то функция описывается и используется как классическая функция типа y=sin(x).

Решение:

Исходные данные: Два операнда x, y.

Результат: Сумма 2 двух значений.

Код программы:

```
//31.03.2020 Сидоров С.С. ИСП-21
//Пример1. Создадим и используем функции для сложения 2 чисел
Ссылок: 0
class Program
{
    //Функция сумма 2 чисел
    //Входные значения: value1, value2 - исходные числа
    //Результат функции сумма чисел
    ссылка: 1
    static int Summ(int value1, int value2)
    {
        int sum;// Сумма чисел
        sum = value1 + value2;
        return sum;//Возврат ответа
    }// end Summ

    // Главная программа
    Ссылок: 0
    static void Main(string[] args)
    {
        int zn1, //Значение 1
            zn2, //Значение 2
            sum; //Сумма чисел
        Console.WriteLine("Введите 2 значения");
        zn1 = Convert.ToInt32(Console.ReadLine());
        zn2 = Convert.ToInt32(Console.ReadLine());
        sum = Summ(zn1, zn2);//Вызов функции
        Console.WriteLine("Сумма значений = " + sum);
        Console.ReadKey();//Пауза
    }// end Main
}
```

Формальные и фактические параметры

Формальные параметры используются при описании параметров (аргументов) функции. В примере 1 – это параметры ***value1*** и ***value2***. Они задаются некоторыми уникальными именами и внутри функции доступны как локальные переменные.

Фактические параметры используются в основной программе. Они используются при вызове функции на месте формальных параметров. В момент вызова функции значения фактических параметров присваиваются формальным параметрам. Соответственно, имена формальных и фактических параметров могут совпадать, а могут не совпадать. В примере 1 – это параметры ***zn1*** и ***zn2***.

Локальные и глобальные переменные

Локальная переменная - переменная, объявленная внутри какой-либо функции или блока. Областью видимости локальных переменных является тело функции или блока, в которой эта переменная объявлена.

Глобальная переменная - переменная, объявленная за пределами всех функций или блоков. Областью видимости глобальных переменных является вся программа.

Обмен данными между программой и функциями

При описании функции указываются формальные параметры, которые указывают, с какими величинами следует обращаться к этой функции. Заголовок функции определяет количество параметров, их последовательность и типы.

Параметры, используемые в функции, могут быть трех видов:

1. ***Входные параметры*** (например: int value), используются для входных данных.

Передаются основной программой в функцию через стек в виде их копий и, следовательно, внутри функции параметр может изменяться, но фактический параметр функцией измениться не может.

В качестве фактического параметра на месте входного параметра при вызове функции может выступать любое выражение совместимого для присваивания типа, переменная или константа.

2. ***Входные параметры. Модификатор in*** (например: in int value), используются для входных данных.

Передаются основной программой в функцию по ссылке, внутри функции значение параметра изменить будет нельзя.

Обычно используется, если переменная имеет большой размер (массив, запись и т.д.), то копия такого параметра займет большую часть стека и даже может его переполнить. Это же приводит и к уменьшению быстродействия программы. В этой ситуации параметр лучше передать через ссылку с защитой от его изменения

3. Выходные параметры. Модификатор out (например: out int result), используются для выходных данных.

Обратите внимание, что функции, использующие такие параметры, обязательно должны присваивать им определенное значение.

При использовании ключевое слово ***out*** используется как при определении функции, так и при её вызове.

При вызове функции на месте выходного параметра в качестве фактического параметра должна использоваться переменная идентичного типа.

4. Передача параметров по ссылке. Модификатор ref (например: ref int valueA), используются для полного управления передаваемых параметров.

При передаче через ссылку функция получает значение, передаваемое через фактический параметр, а также возвращает в фактический параметр внесенные изменения, т.е. может играть роль входного/ выходного параметра.

При использовании ключевое слово ***ref*** используется как при определении функции, так и при её вызове.

При вызове функции на месте параметра в качестве фактического параметра должна использоваться переменная идентичного типа.

Возврат значения и завершение работы функции

Для завершения работы функции может и должна использоваться инструкция ***return***. Она завершает выполнение функции. После этого управление возвращается той функции, из которой была вызвана данная. Инструкция ***return*** может употребляться в двух формах:

```
return;
return выражение;
```

Первая форма используется в функциях, для которых типом возвращаемого значения является ***void***. Использовать ***return*** в таких случаях обязательно, если нужно принудительно завершить работу. Такое применение ***return*** напоминает инструкцию ***break***.

Во второй форме инструкции ***return*** указывается то значение, которое функция должна вернуть. Это значение может быть сколь угодно сложным выражением, даже содержать вызов функции. В функции, не объявленная с ***void*** в качестве типа возвращаемого значения, обязательно использовать вторую форму ***return***, иначе произойдет ошибка компиляции.

Пример 2

Даны стороны прямоугольника, найти площадь и периметр.

Выбор метода решения:

В данном случае т.к. результатов функции два, то возврат значений функции происходит через аргументы функции PlPer(x, y, Pl, Per).

Решение:

Исходные данные: Два операнда x, y стороны прямоугольника.

Результат: pl - площадь per - периметр

Код программы

```
//31.03.2020 Сидоров С.С. ИСП-21
//Пример2. Даны стороны прямоугольника, найти площадь и периметр
Ссылок: 0
class Program
{
    //Функция нахождения площади и периметра
    //Входные значения: sideA, sideB - стороны прямоугольника
    //Выходные значения: area - площадь perimeter - периметр
    ссылка: 1
    static void AreaPerimetr(int sideA, int sideB, out int area, out int perimeter)
    {
        area = sideA * sideB;
        perimeter = (sideA + sideB) * 2;
    }// end AreaPerimetr

    // Главная программа
    Ссылок: 0
    static void Main(string[] args)
    {
        int zn1, zn2, // Стороны прямоугольника
            pl, // Площадь
            per; // Периметр
        Console.WriteLine("Введите стороны прямоугольника");
        zn1 = Convert.ToInt32(Console.ReadLine());
        zn2 = Convert.ToInt32(Console.ReadLine());
        AreaPerimetr(zn1, zn2, out pl, out per);//Вызов функции
        Console.WriteLine("Площадь = " + pl);
        Console.WriteLine("Периметр = " + per);
        Console.ReadKey();//Пауза
    } // end Main
}
```

Параметры со значениями по умолчанию (необязательные параметры)

Для ряда параметров функции на этапе её создания могут быть указаны значения по умолчанию, эти значения будут автоматически подставляться, если при вызове явно не было задано значение параметров.

Параметры по умолчанию удобно использовать, когда для функции заранее ясно, с какими параметрами её будут вызывать чаще всего.

```
int pow(int a, int p = 2) {

...
}
```

Теперь можно вызывать функцию, не указывая последнего параметра:

```
cout << pow(5); // 25
```

Что равносильно вызову:

```
cout << pow(5,2); // 25
```

Перегрузка функций

Уникальность функции определяется не только её именем, но и набором её параметров и типом возвращаемого значения. C# позволяет создавать функцию с именем ранее существовавшей при условии, что у новой функции будет иной набор параметров. Таким образом, перегрузка функций нужна для того, чтобы избежать дублирования имён функций, выполняющих сходные действия, но с различной программной логикой. Например, нам нужно складывать два числа, а числа бывают целые и вещественные, следовательно, нам нужно иметь две разные функции для разных типов чисел типа ***SumInt*** и ***SumDouble***. Перегрузка позволяет иметь две функции с одним названием ***Sum*** для сложения, как целых, так и вещественных чисел, см. пример ниже.

```
int Sum(int value1, int value2){
      int rez;
      rez = value1 + value2;
      return rez;
}

double Sum(double value1, double value2){
      double rez;
      rez = value1 + value2;
      return rez;
}
```

Пример 3

Описать функцию DigitCoun5(K, C, P), находящую количество C цифр целого положительного числа K, а количество цифр пять P (K — входной, C и P — выходные параметры целого типа). С помощью этой процедуры найти количество цифр и количество цифр пять для каждого из пяти данных целых чисел.

Выбор метода решения:

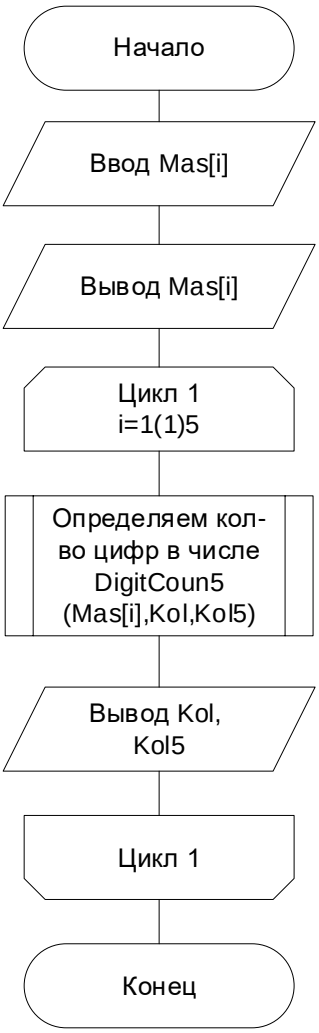
От целого числа с конца отделяем по одной цифре и считаем их количество. Одновременно с этим проверяем отделенные цифры, определяя из них цифры 5, и считаем их количество. Процесс продолжаем до тех пор, пока в исходном числе есть цифры. Для отделения цифр с конца числа используем операции деления / и %.

Распишем эти действия более подробно.

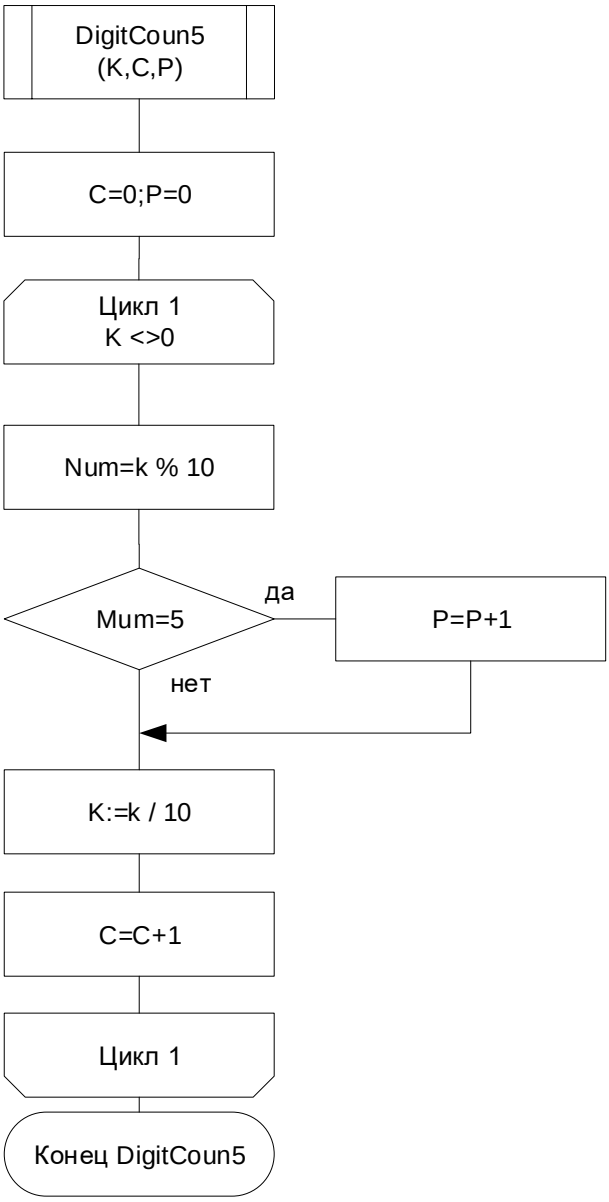
1. В этой задаче имеем один входной и два выходных параметров. Входной параметр описываем как параметр значение, выходные как параметры-переменные.
2. **Алгоритм** процедуры будет следующим (процедуру назовем DigitCoun5):
3. Пока число содержит цифры **While (k <> 0) do**.
4. Получаем цифру с конца Num=k **%** 10.
5. Если это цифра 5 увеличиваем счетчик цифр 5 **If** Num = 5 **тогда** P=P+1.
6. Увеличиваем общий счетчик цифр C=C+1;
7. Убираем цифру с конца числа k=k / 10;

Блок-схема алгоритма

Основная программа



Подпрограмма DigitCoun5



Код программы

```
//31.03.2020 Сидоров С.С. ИСП-21
//Пример3. Описать функцию DigitCoun5(K, C, P), находящую количество
//С цифр целого положительного числа K, а количество цифр пять P
//(K – входной, C и P – выходные параметры целого типа). С помощью
//этой процедуры найти количество цифр и количество цифр пять для
//каждого из пяти данных целых чисел.
Ссылок: 0
class Program
{
    // Определение кол-ва цифр и кол-ва 5 в числе
    // Входные данные: K - целое число
    // Выходные данные: C - кол-во цифр в числе
    // P - кол-во цифр 5 в числе
    ссылка: 1
    static void DigitCoun5(int k, out int c, out int p)
    {
        int num; //Цифра в числе
        c = 0; p = 0; // Обнуляем счетчики
        while (k != 0) // Пока цифры есть в числе
        {
            num = k % 10;       // получем цифру с конца числа
            if (num == 5) p = p + 1; // Если эта цифра 5, то счетчик +1
            k = k / 10;         // Удалям цифру с конца числа
            c = c + 1;          // Счетчик цифр увеличиваем на 1
        }
    }

//Главная программа
Ссылок: 0
static void Main(string[] args)
{
    int Kol, //Кол-во цифр в числе
        Kol5, //Кол-во 5 в числе
        i;
    int[] Mas = new int[5]; // Массив чисел
    //Ввод исходных данных
    Console.WriteLine("Введите 5 чисел");
    for (i = 0; i < Mas.Length; i++)
    {
        Console.Write("Введите число " + i + " =>");
        Mas[i]=Convert.ToInt32(Console.ReadLine());
    }

    //Вывод исходных данных
    Console.WriteLine("Исходные значения");
    for (i = 0; i < Mas.Length; i++) Console.Write(Mas[i] + " ");
    Console.WriteLine();

    //Расчет кол-во цифр в каждом числе и кол. цифр 5 и вывод результата
    Console.WriteLine("Общее кол-во цифр и цифр 5 для каждого числа");
    for (i = 0; i < Mas.Length; i++)
    {
        DigitCoun5(Mas[i], out Kol, out Kol5);
        Console.WriteLine("В числе {0,5} {1} цифр и {2} пятерок", Mas[i], Kol, Kol5);
    }
    Console.ReadKey();//Пауза

}
```

Практическая работа №14.

Организация функций

Задание №1. Составьте алгоритм и программу по заданию.

1. Описать функцию Sign(X) целого типа, возвращающую для вещественного числа X следующие значения: –1, если X < 0; 0, если X = 0; 1, если X > 0. С помощью этой функции найти значение выражения Sign(A) + Sign(B) для данных вещественны чисел A и B.

2. Описать функцию RootsCount(A, B, C) целого типа, определяющую количество корней квадратного уравнения $A \cdot x^2 + B \cdot x + C = 0$ (A, B, C — вещественные параметры, A . 0). С ее помощью найти количество корней для каждого из трех квадратных уравнений с данными коэффициентами. Количество корней определять по значению дискриминанта: $D = B^2 – 4 \cdot A \cdot C$.

3. Описать функцию CircleS(R) вещественного типа, находящую площадь круга радиуса R (R — вещественное). С помощью этой функции найти площади трех кругов с данными радиусами. Площадь круга радиуса R вычисляется по формуле $S = 3.14 \cdot R^2$. В качестве значения использовать 3.14.

4. Описать функцию RingS(R1, R2) вещественного типа, находящую площадь кольца, заключенного между двумя окружностями с общим центром и радиусами R1 и R2 (R1 и R2 — вещественные, R1 > R2). С ее помощью найти площади трех колец, для которых даны внешние и внутренние радиусы. Воспользоваться формулой площади круга радиуса R: $S = 3.14 \cdot R^2$. В качестве значения . использовать 3.14.

5. Описать функцию TriangleP(a, h), находящую периметр равнобедренного треугольника по его основанию a и высоте h, проведенной к основанию (a и h — вещественные). С помощью этой функции найти периметры трех треугольников, для которых даны основания и высоты. Для нахождения боковой стороны b треугольника использовать теорему Пифагора: $b^2 = (a/2)^2 + h^2$.

6. Описать функцию SumRange(A, B) целого типа, находящую сумму всех целых чисел от A до B включительно (A и B — целые). Если A > B, то функция возвращает 0. С помощью этой функции найти суммы чисел от A до B и от B до C, если даны числа A, B, C.

7. Описать функцию Calc(A, B, Op) вещественного типа, выполняющую над ненулевыми вещественными числами A и B одну из арифметических операций и возвращающую ее результат. Вид операции определяется целым параметром Op: 1 — вычитание, 2 — умножение, 3 — деление, остальные значения — сложение. С помощью Calc выполнить для данных A и B операции, определяемые данными целыми N1, N2, N3.

8. Описать функцию Quarter(x, y) целого типа, определяющую номер координатной четверти, в которой находится точка с ненулевыми вещественными координатами (x, y). С помощью этой функции найти номера координатных четвертей для трех точек с данными ненулевыми координатами.

9. Описать функцию Even(K) логического типа, возвращающую True, если целый параметр K является четным, и False в противном случае. С ее помощью найти количество четных чисел в наборе из 10 целых чисел.

10. Описать функцию IsSquare(K) логического типа, возвращающую True, если целый параметр K (> 0) является квадратом некоторого целого числа, и False в противном случае. С ее помощью найти количество квадратов в набореиз 10 целых положительных чисел.

11. Описать функцию IsPower5(K) логического типа, возвращающую True, если целый параметр K (> 0) является степенью числа 5, и False в противном случае. С ее помощью найти количество степеней числа 5 в наборе из 10 целых положительных чисел.

12. Описать функцию IsPowerN(K, N) логического типа, возвращающую True, если целый параметр K (> 0) является степенью числа N (> 1), и False в противном случае. Дано число N (> 1) и набор из 10 целых положительных чисел. С помощью функции IsPowerN найти количество степеней числа N в данном наборе.

13. Описать функцию IsPrime(N) логического типа, возвращающую True, если целый параметр N (> 1) является простым числом, и False в противном случае (число, большее 1, называется простым, если оно не имеет положительных делителей, кроме 1 и самого себя). Дан набор из 10 целых чисел, больших 1. С помощью функции IsPrime найти количество простых чисел в данном наборе.

14. Описать функцию DigitCount(K) целого типа, находящую количество цифр целого положительного числа K. Используя эту функцию, найти количество цифр для каждого из пяти данных целых положительных чисел.

15. Описать функцию DigitN(K, N) целого типа, возвращающую N-ю цифру целого положительного числа K (цифры в числе нумеруются справа налево). Если количество цифр в числе K меньше N, то функция возвращает –1. Для каждого из пяти данных целых положительных чисел K1, K2, …, K5 вызвать функцию DigitN с параметром N, изменяющимся от 1 до 5.

16. Описать функцию DegToRad(D) вещественного типа, находящую величину угла в радианах, если дана его величина D в градусах (D — вещественное число, 0 < D < 360). Воспользоваться следующим соотношением: 180° = . радианов. В качестве значения. использовать 3.14. С помощью функции DegToRad перевести из градусов врадианы пять данных углов.

17. Описать функцию RadToDeg(R) вещественного типа, находящую величину угла в градусах, если дана его величина R в радианах (R — вещественное число, 0 < R < 2·.). Воспользоваться следующим соотношением: 180° = . радианов. В качестве значения использовать 3.14. С помощью функции RadToDeg перевести из радианов в градусы пять данных углов.

18. Описать функцию Fact(N) вещественного типа, вычисляющую значение факториала N! = 1·2·…·N (N > 0 — параметр целого типа; вещественное возвращаемое значение используется для того, чтобы избежать целочисленного переполнения при больших значениях N). С помощью этой функции найти факториалы пяти данных целых чисел.

Задание №2. Составьте алгоритм и программу по заданию.

1. Описать процедуру PowerA3(A, B), вычисляющую третью степень числа A и возвращающую ее в переменной B (A — входной, B — выходной параметр; оба параметра являются вещественными). С помощью этой процедуры найти третьи степени пяти данных чисел.

2. Описать процедуру PowerA234(A, B, C, D), вычисляющую вторую, третью и четвертую степень числа A и возвращающую эти степени соответственно в переменных B, C и D (A — входной, B, C, D — выходные параметры; все параметры являются вещественными). С помощью этой процедуры найти вторую, третью и четвертую степень пяти данных чисел.

3. Описать процедуру Mean(X, Y, AMean, GMean), вычисляющую среднее арифметическое AMean = (X + Y)/2 и среднее геометрическое GMean = $\sqrt{X*Y}$ двух положительных чисел X и Y (X и Y — входные, AMean и GMean — выходные параметры вещественного типа). С помощью этой процедуры найти среднее арифметическое и среднее геометрическое для пар (A, B), (A, C), (A, D), если даны A, B, C, D.

4. Описать процедуру TrianglePS(a, P, S), вычисляющую по стороне a равностороннего треугольника его периметр P = 3·a и площадь S = $a^2\sqrt{3/4}$ (a — входной, P и S — выходные параметры; все параметры являются вещественными). С помощью этой процедуры найти периметры и площади трех равносторонних треугольников сданными сторонами.

5. Описать процедуру SortInc3(A, B, C), меняющую содержимое переменных A, B, C таким образом, чтобы их значения оказались упорядоченными по возрастанию (A, B, C — вещественные параметры, являющиеся одновременно входными и выходными). С помощью этой процедуры упорядочить по возрастанию два данных набора из трех чисел: (A1, B1, C1) и (A2, B2, C2).

6. Описать процедуру RectPS(x1, y1, x2, y2, P, S), вычисляющую периметр P и площадь S прямоугольника со сторонами, параллельными осям координат, по координатам (x1, y1), (x2, y2) его противоположных вершин (x1, y1, x2, y2 — входные, P и S — выходные параметры вещественного типа). С помощью этой процедуры найти периметры и площади трех прямоугольников с данными противоположными вершинами.

7. Описать процедуру DigitCountSum(K, C, S), находящую количество C цифр целого положительного числа K, а также их сумму S (K — входной, C и S — выходные параметры целого типа). С помощью этой процедуры найти количество и сумму цифр для каждого из пяти данных целых чисел.

8. Описать процедуру InvertDigits(K), меняющую порядок следования цифр целого положительного числа K на обратный (K — параметр целого типа, являющийся одновременно входным и выходным). С помощью этой процедуры поменять порядок следования цифр на обратный для каждого из пяти данных целых чисел.

9. Описать процедуру AddRightDigit(D, K), добавляющую к целому положительному числу K справа цифру D (D — входной параметр целого типа, лежащий в диапазоне 0–9, K — параметр целого типа, являющийся одновременно входным и выходным). С помощью этой процедуры последовательно добавить к данному числу K справа данные цифры D1 и D2, выводя результат каждого добавления.

10. Описать процедуру ShiftLeft3(A, B, C), выполняющую левый циклический сдвиг: значение A переходит в C, значение C — в B, значение B — в A (A, B, C — вещественные параметры, являющиеся одновременно входными и выходными). С помощью этой процедуры выполнить левый циклический сдвиг для двух данных наборов из трех чисел: (A1, B1, C1) и (A2, B2, C2).

11. Описать процедуру Swap(X, Y), меняющую содержимое переменных X и Y (X и Y — вещественные параметры, являющиеся одновременно входными и выходными). С ее помощью для данных переменных A, B, C, D последовательно поменять содержимое следующих пар: A и B, C и D, B и C и вывести новые значения A, B, C, D.

12. Описать процедуру Minmax(X, Y), записывающую в переменную X минимальное из значений X и Y, а в переменную Y — максимальное из этих значений (X и Y — вещественные параметры, являющиеся одновременно входными и выходными). Используя четыре вызова этой процедуры, найти минимальное и максимальное из данных чисел A, B, C, D.

13. Описать процедуру SortDec3(A, B, C), меняющую содержимое переменных A, B, C таким образом, чтобы их значения оказались упорядоченными по убыванию (A, B, C — вещественные параметры, являющиеся одновременно входными и выходными). С помощью этой процедуры упорядочить по убыванию два данных набора из трех чисел: (A1, B1, C1) и (A2, B2, C2).

14. Описать процедуру ShiftRight3(A, B, C), выполняющую правый циклический сдвиг: значение A переходит в B, значение B — в C, значение C — в A (A, B, C — вещественные параметры, являющиеся одновременно входными и выходными). С помощью этой процедуры выполнить правый циклический сдвиг для двух данных наборов из трех чисел: (A1, B1, C1) и (A2, B2, C2).

15. Описать процедуру PowerA35(A, B, C), вычисляющую третью и пятую степень числа A и возвращающую ее переменным B и C(A — входной, B, C — выходной параметр; все параметры являются вещественными). С помощью этой процедуры найти третьи и пятые степени пяти данных чисел.

16. Описать процедуру RectPS(a, P, S), вычисляющую периметр P и площадь S квадрата со стороной a, (a — входные, P и S — выходные параметры вещественного типа). С помощью этой процедуры найти периметры и площади трех квадратов с данными противоположными вершинами.

17. Описать процедуру AddLeftDigit(D, K), добавляющую к целому положительному числу K слева цифру D (D — входной параметр целого типа, лежащий в диапазоне 1–9, K — параметр целого типа, являющийся одновременно входным и выходным). С помощью этой процедуры последовательно добавить к данному числу K слева данные цифры D1 и D2, выводя результат каждого добавления.

18. Описать процедуру DigitCountMax(K, C, M), находящую количество C цифр целого положительного числа K, а также максимальную цифру M этого числа(K — входной, C и M — выходные параметры целого типа). С помощью этой процедуры найти количество и максимальную цифру для каждого из пяти данных целых чисел.

Контрольные вопросы

1. С какой целью оформляются и где описываются процедуры и функции?
2. В каком случае подпрограмма оформляется как процедура и в каком случае как функция?
3. Каков синтаксис описания процедуры? Какой вид имеет заголовок процедуры?
4. Каков синтаксис описания функции?
5. Каковы особенности реализации функций?
6. Как выполняются обращения к процедуре и к функции?
7. Какие переменные подпрограмм называются локальными и какова область их действия? Какие переменные называются глобальными?
8. Какие параметры используются для передачи данных между подпрограммами, какими способами осуществляется эта передача?
9. Что такое формальные параметры подпрограммы и как они описываются? Что такое фактические параметры подпрограммы?
10. Какие существуют виды формальных параметров?
11. Как передаются в подпрограмму параметры-значения, что может использоваться в качестве фактического параметра для параметра-значения?
12. Как передаются в подпрограмму параметры-переменные, что может использоваться в качестве фактического параметра для параметра-переменной?
13. Как передаются в подпрограмму параметры-константы, что может использоваться в качестве фактического параметра для параметра-константы?
14. Каким образом можно осуществить досрочный выход из подпрограммы?
15. Существуют ли подпрограммы без параметров?
16. Какое количество значений возвращает функция?
17. Как определить тип значения, возвращаемый функцией?
18. Могут ли фактические параметры быть выражениями?
19. Сколько элементов должен содержать список фактических параметров?
20. Сколько элементов может содержать список формальный параметров?
21. Каково соответствие между фактическими и формальными параметрами?
22. Могут ли фактические параметры быть именами переменных?
23. Какие переменные называются локальными?
24. Что такое область видимости переменной?
25. Может ли имя локальной переменной совпадать с глобальной?
26. Может ли в основной программе функция вызываться внутри выражений?
27. Наличие какого оператора необходимо для возвращения значения из функции в вызывающую программу?