ЛАБОРАТОРНАЯ РАБОТА № 23

РЕКУРСИВНЫЕ ФУНКЦИИ

Цель работы: получение навыков в написании программ с использованием рекурсивных функций.

Краткие теоретические сведения

Рекурсивной (самовызываемой или самовызывающей) называют функцию, которая прямо или косвенно вызывает сама себя.

При каждом обращении к рекурсивной функции создается новый набор объектов автоматической памяти, локализованных в коде функции.

Возможность прямого или косвенного вызова позволяет различать прямую или косвенную рекурсии. Функция называется косвенно рекурсивной в том случае, если она содержит обращение к другой функции, содержащей прямой или косвенный вызов первой функции. В этом случае по тексту определения функции ее рекурсивность (косвенная) может быть не видна. Если в функции используется вызов этой же функции, то имеет место прямая рекурсия, т.е. функция по определению рекурсивная.

Рекурсивные алгоритмы эффективны в задачах, где рекурсия использована в самом определении обрабатываемых данных. Поэтому изучение рекурсивных методов нужно проводить, вводя динамические структуры данных с рекурсивной структурой. Рассмотрим вначале только принципиальные возможности, которые предоставляет язык Си для организации рекурсивных алгоритмов.

В рекурсивных функциях необходимо выполнять следующие правила.

1. При каждом вызове в функцию передавать модифицированные данные.

2. На каком-то шаге должен быть прекращен дальнейший вызов этой функции, это значит, что рекурсивный процесс дол-

жен шаг за шагом упрощать задачу так, чтобы для нее появилось нерекурсивное решение, иначе функция будет вызывать себя бесконечно.

3. После завершения очередного обращения к рекурсивной функции в вызывающую функцию должен возвращаться некоторый результат для дальнейшего его использования.

Пример 1. Заданы два числа *a* и *b*, большее из них разделить на меньшее, используя рекурсию.

Текст программы может быть следующим:

```
        . . .
double proc(double, double);
void main (void)
{
        double a,b;
        puts(" Введи значения a, b : ");
        scanf("%lf %lf", &a, &b);
        printf("\n Результат деления : %lf", proc(a,b));
}
//——————————————— Функция ———————————————
double proc( double a, double b)   {
        if ( a< b ) return proc ( b, a );
                else  return a/b;
}
```

Если *a* больше *b,* условие, поставленное в функции, не выполняется и функция *proc* возвращает нерекурсивный результат.

Пусть теперь условие выполнилось, тогда функция *proc* обращается сама к себе, аргументы в вызове меняются местами и последующее обращение приводит к тому, что условие вновь не выполняется и функция возвращает нерекурсивный результат.

Пример 2. Функция для вычисления факториала *неотрицательного* значения *k* (для возможных отрицательных значений необходимо добавить дополнительные условия).

```
double fact (int k) {
        if ( k < 1 ) return 1;
        else
                return  k * fact ( k – 1);
}
```

Для нулевого значения параметра функция возвращает 1 (0! = 1), в противном случае вызывается та же функция с уменьшенным на 1 значением параметра и результат умножается на текущее значение параметра. Тем самым для значения параметра *k* организуется вычисление произведения

$$k \ * \ (k–1) \ * \ (k–2) \ * ... * \ 3 \ * \ 2 \ * \ 1 \ * 1$$

Последнее значение «1» – результат выполнения условия $k < 1$ при $k = 0$, т.е. последовательность рекурсивных обращений к функции *fact* прекращается при вызове *fact*(0). Именно этот вызов приводит к последнему значению «1» в произведении, так как последнее выражение, из которого вызывается функция, имеет вид: 1 * *fact*(1 – 1).

Порядок выполнения работы

1. Изучить теоретические сведения.
2. Ответить на контрольные вопросы.
3. Выполнить задание.

Контрольные вопросы

1. Что такое рекурсивная функция?
2. Что такое прямая рекурсия?
3. Что такое косвенная рекурсия?
4. На каком-то шаге должен быть прекращен дальнейший вызов рекурсивной функции?

Задания для выполнения

Написать программы решения следующих задач, используя рекурсивную функцию.

1. Найти наименьшую цифру в десятичной записи заданного натурального числа.
2. Подсчитать количество цифр в заданном натуральном числе.
3. Вычислить наибольший общий делитель двух натуральных чисел.
4. Найти число, которое образуется из заданного натурального числа при записи его цифр в обратном порядке. Например, для числа 1234 получаем результат 4321.
5. Вычислить сумму: 1! + 2! + 3! + … +n! ($n \leq 15$).

6. Вычислить сумму: $2! + 4! + 6! + \ldots + n!$ ($n \le 16$, n – четное).

7. Логическая функция возвращает 1, если ее аргумент – простое число.

8. Вычислить функцию Аккермана для всех неотрицательных целых аргументов m и n:

$$A(m,n) = \begin{cases} A(0,n) = n+1, \\ A(m,0) = A(m-1,1),\ m>0, \\ A(m,n) = A(m-1, A(m,n-1)),\ m,n>0 \end{cases}$$

9. Найти количество нечетных цифр в десятичной записи заданного натурального числа.

10. Найти количество цифр, кратных 3, в десятичной записи заданного натурального числа.

11. Вычислить значение $C_n^k = \dfrac{n!}{k!(n-k)!}$ (значение 0!=1).

12. Вычислить произведение четного количества n ($n \ge 2$) сомножителей следующего вида:

$$y = \left(\frac{2}{1}\cdot\frac{2}{3}\right)\cdot\left(\frac{4}{3}\cdot\frac{4}{5}\right)\cdot\left(\frac{6}{5}\cdot\frac{6}{7}\right)\cdot \ldots\ .$$

13. Вычислить $y = x^n$ по следующему правилу: $y = (x^{n/2})^2$, если n четное и $y = x \cdot y^{n-1}$, если n нечетное.

14. Вычислить значение $y(n) = \sqrt{1+\sqrt{2+\ldots+\sqrt{n}}}$.

15. Вычислить значение $x = \sqrt{a}$, используя рекуррентную формулу $x_n = = \dfrac{1}{2}\left(x_{n-1} + \dfrac{a}{x_{n-1}}\right)$, в качестве начального значения использовать $x_0 = 0{,}5\cdot(1 + a)$.