Тема 1.7 Метод пошаговой детализации (последовательного уточнения) разработки алгоритмов. Особенности использования массивов в качестве параметров. Обработка двумерных массивов

В большинстве случаев при разработке алгоритма не удается сразу получить удовлетворительный результат, поэтому составление алгоритма проводится методом проб и устранения ошибок и для получения окончательного результата требуется несколько шагов коррекции и анализа. Как правило, процесс разработки алгоритма проходит несколько шагов детализации. Первоначально строится укрупненная схема алгоритма, в которой отражаются наиболее важные и существенные связи между исследуемыми процессами или частями процесса. На последующих этапах раскрываются, т.е. детализируются выделенные ранее части вычислительного процесса, которые имеют некоторые самостоятельные значения. Кроме того, на каждом этапе детализации выполняется многократная проверка и исправление схемы алгоритма. Подобный подход позволяет избежать возможных ошибочных решений.

Виды алгоритмов:

- линейные алгоритмы;
- разветвляющиеся алгоритмы;
- циклические алгоритмы.

Линейным называется алгоритм, в котором все этапы решения задачи выполняются строго последовательно.

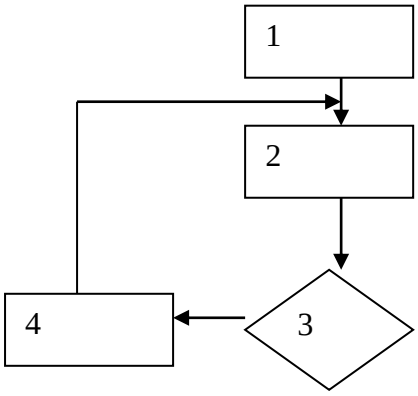
Пример:

Поменять местами a и b.

1 способ	2 способ
c:=a	a=a+b
a:=b	b=a-b
b:=c	a=a-b

Разветвляющимся называется такой алгоритм, в котором выбирается один из нескольких возможных путей вычислительного процесса. Условием разветвляющегося алгоритма является наличие операции проверки условия.

Циклическим называется алгоритм, в котором получение результата обеспечивается многократным выполнением одних и тех же операций. Структуру любого циклического алгоритма может определить следующая схема:



1 – присваивание переменным начальных значений
2 – вычисление результата
3 – проверка условия окончания цикла
4 – изменение переменных

Особенности использования массивов в качестве параметров функций

Когда массив используется в качестве аргумента функции, передается только адрес массива, а не копия всего массива. При вызове функции с именем массива в функцию передается указатель на первый элемент массива. Параметр должен иметь тип совместимый с указателем.

Имеется три способа объявления параметра, предназначенного для получения указателя на массив.

- объявление как массив

```
#include <stdio.h>

void display (int num[10]);

int main ()
{
int t[10], i;
for (i=0; i<10; i++)
t[i]=i;
display(t);
return 0;
}

void display (int num[10])
{
int i;
for (i=0; i<10; i++)
printf("%d", num[i]);
}
```

Хотя параметр num объявлен как целочисленный массив из 10 элементов, С автоматически преобразует его к целочисленному указателю, поскольку не существует параметра, который на самом деле мог бы принять массив. Передается только указатель на массив, поэтому должен быть параметр, способный принять его.

- объявление параметра для указания на безразмерный массив

```
void display (int num[])
{
int i;
for (i=0; i<10; i++)
```

```
printf(“%d”, num[i]);
}
```

где num объявлен как целочисленный массив неизвестного размера. Поскольку С не предоставляет проверку границ массива, настоящий размер массива не имеет никакого отношения к параметру. Данный пример также определяет num как целочисленный указатель.

- через указатель (типичный способ при написании профессиональных программ)

```
void display (int *num)
{
int i;
for (i=0; i<10; i++)
printf(“%d”, num[i]);
}
```

Важно понять, что при использовании массива в качестве аргумента функции происходит передача в функцию его адреса. Это означает, что код внутри функции действует и может изменять настоящее значение массива, используемого при вызове.

Передача многомерного массива

Многомерный массив также передается как указатель на его первый элемент. В тоже время поскольку элементами многомерного массива являются другие массивы, то указатель на первый элемент многомерного массива фактически будет представлять указатель на массив.

При определении параметра как указателя на массив размер второй размерности (а также всех последующих) должен быть определен, так как данный размер является частью типа элемента.

*void print (int (*num) [3])* – предполагается, что передаваемый массив будет двумерным, и все его подмассивы будут иметь по три элемента. Стоит обратить внимание на скобки внутри имени параметра, которые и позволяют определить параметр как указатель на массив.

*void print (int *num [3])* – в данном случае параметр определен как массив указателей, а не как указатель на массив

```
...
void print (int (*) [3], int);
```

```
int main ()
{
int table [3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
int rowCount = sizeof(table)/sazeof(table[0]);
print (table, rowCount);
return 0;
}

void print (int (*num) [3], int rowCount)
{
      int columnsCount = sizeof(*num)/sazeof(*num[0]);
      for (int i=0; i<rowCount; i++)
        { for (int j=0; j< columnsCount; j++)
              {
              cout<<num[i][j]<<"\t";
              }
  cout <<endl;
          }
}
```