Tassadaq Hussain

Recursive Functions

Recursion

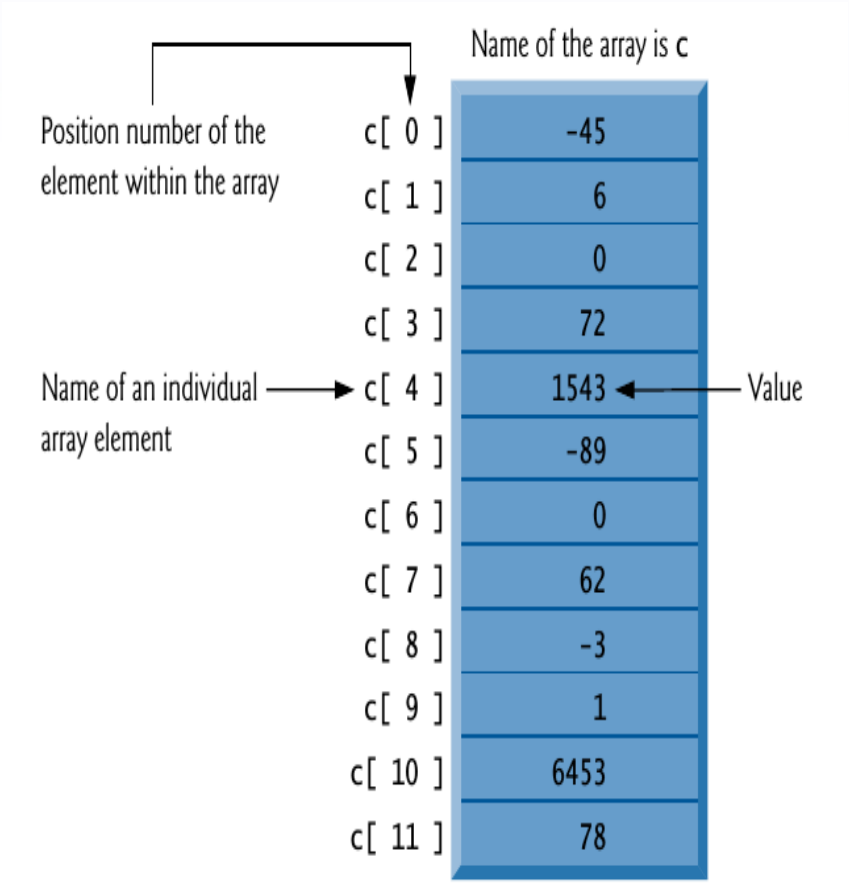
For some problems, it's useful to have functions call themselves. A recursive function is a function that calls itself, either directly, or indirectly (through another function).

```
#include <iostream>
#include <iomanip>
#include <stdio.h>
using namespace std;
// recursive definition of function factorial
int factorial(int number )
{
if ( number <= 1 ) // test for base case
return 1; // base cases: 0! = 1 and 1! = 1
else // recursion step
return number * factorial( number - 1 );
} // end function factorial

int main()
{
// calculate the factorials of 0 through 10
for ( int counter = 0; counter <= 10; ++counter )
cout<<" Factorial of   "<<counter << " = " << factorial(counter)<<endl;
//cout<< "   " << counter << "! = " << factorial( counter) << endl;
} // end main
```

Arrays and Vectors

Arrays are data structures consisting of related data items of the same type.

Array of 12 Elements C[12]



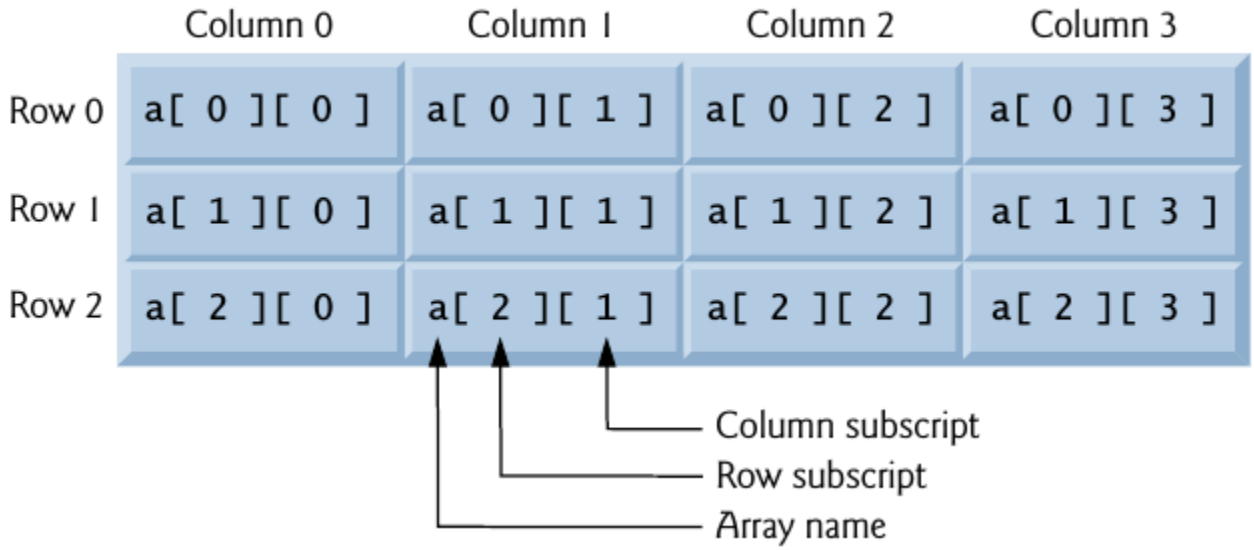

```
int c[12] = { -45 ,6, 0, 72, 1543,
    -89, 0, 62, -3, 1, 6453, 78};
```

Example Using an Array

```
#include <iostream>
#include <iomanip>
using namespace std;

int main()
{
   int n[ 10 ]; // n is an array of 10 integers

   // initialize elements of array n to 0
   for ( int i = 0; i < 10; ++i )
      n[ i ] = 0; // set element at location i to 0

   cout << "Element" << setw( 13 ) << "Value" << endl;

   // output each array element's value
   for ( int j = 0; j < 10; ++j )
      cout << setw( 7 ) << j << setw( 13 ) << n[ j ] << endl;
} // end main
```

Multidimensional Arrays

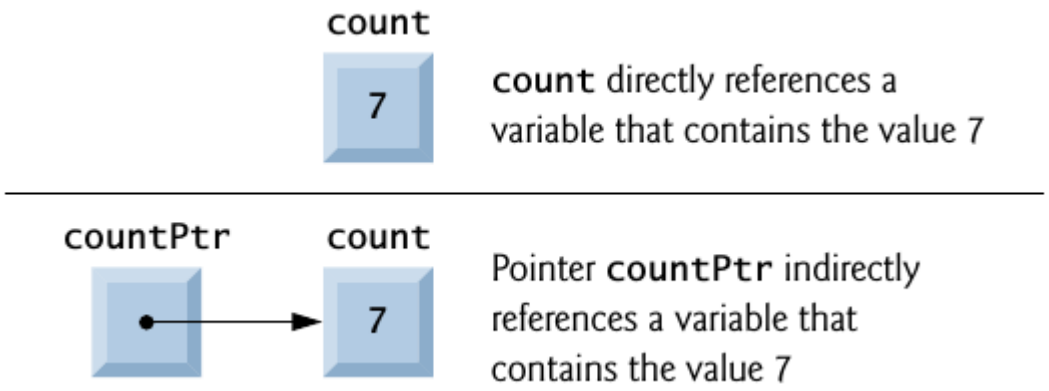
Int name_array[row][column]




```
int array1[ rows ][ columns ] = { { 1, 2, 3 }, { 4, 5, 6 } };
```

Pointer

Pointer variables contain memory addresses as their values. contains a specific value. A pointer contains the memory add



int y = 5; // declare variable y
int *yPtr; // declare pointer variable yPtr

```
yPtr = &y; // assign address of y to yPtr
```