Monday, January 27, 2003

MAT 102 - Data Structures

Arrays, Structures, and Project #1
The Array

- used to store a collection of items of the same type
- provides quick access to any element
- inserting an element in the middle of an array is costly
- the size of an array is set when the array is defined
- C++ doesn’t check to see if your index is out of range--this can lead to strange errors
```cpp
#include <iostream>
using namespace std;

int main()
{
    int scores[10];
    // index ranges from 0 to 9
    int i;
    for(i=0; i<10; i++)
    {
        cout << "Enter next score " << endl;
        cin >> scores[i];
    }
    for(i=0; i<10; i++)
    {
        cout << scores[i] << " is [" << i << "]" << endl;
    }
    return 0;
}
```
int sort(int a[], int size) { // passing an array to a function

    for (int i=0; i< size; i++)
        cout << "Element " << i << " is " << a[i] << endl;
}

// note that the elements of an array can be changed

int main() {

    int display(int a[], int size) {

        display(scores, 10);
        ...
        int scores[10]
    }

    int main()

    int sort(int a[], int size) {

        passing an array to a function
    }
A C++ structure collects different types of variables into one object.

Useful when organizing data that is related.

Can be passed to a function as one object.

Must be passed by reference if changes are going to made to the variables in a structure.

The Structure
```cpp
#include <iostream>
#include <string>
using namespace std;

struct Address {
    string street;
    string city;
    string state;
    int zip;
};

int main() {
    ...
}

() main

{
    int zip
    string state
    string city
    string street
}
struct Address

using namespace std;
#include <string>
#include <iostream>
#include <sstream>
```
int main() {
    Address address;
    cout << "Enter street address: " << endl;
    getline(cin, address.street);
    cout << "Enter city: " << endl;
    getline(cin, address.city);
    cout << "Enter state: " << endl;
    getline(cin, address.state);
    cout << "Enter zip code: " << endl;
    getline(cin, address.zip);
    return 0;
}
Enter street address: 1600 Pennsylvania Avenue
Enter city: Washington
Enter state: DC
Enter zip code: 10001

The address you entered is:
1600 Pennsylvania Avenue, Washington, DC 10001
Passing variables by reference

- When you pass a variable to a function, the function makes a copy of the variable to work on the copy, not the original variable. To pass by reference, simply put a & in front of the variable name in the function prototype and in the function definition.
- To pass by reference, simply put a & in front of the variable name in the function prototype and in the function definition.
- If you want to make changes to a variable with a reference, you must pass the variable by reference.
- If you want to make changes to a variable, you must pass the variable by reference.
- If you want to make changes to a variable, you must pass the variable by reference.
- This means that the function can only read the variable, it can't make changes.
- This means that the function can only read the variable, it can't make changes.
- When you pass a variable to a function, the function makes a copy of the variable to work on the copy, not the original variable.
int main()
{
    int n = 0;
    cout << n << endl;
    addFive(n);
    cout << n << endl;
    return 0;
}

int addFive(int &var)
{
    var = var + 5;
    return 0;
}
Hints for Project #1

- Use a structure with an array of integers for the digits and an integer for the number of digits.
- Write a function that gets input character by character.
- Use the cin.get() and next() functions to get input.
- Use a structure with an array of integers for the adding function, think very carefully about all the steps that you would do if you were adding two numbers by hand.
- For the adding function, think very carefully about all the steps that you would do if you were adding two numbers by hand.

Hints for Project #1


```cpp
#include <iostream>
#include <cctype>
using namespace std;

const int MAX_SIZE = 100;

struct BigInt
{
  int digits[MAX_SIZE];

  ...

  char operation;
  BigInt num1, num2, num3;
};

int displayResults(BigInt num1, BigInt num2, BigInt num3)
{
  int digitsResults[MAX_SIZE];
  int addNumbers(BigInt num1, BigInt num2, BigInt num3);
  int getInputs(BigInt num1, char operation, BigInt num2);
};

int main()
{
  BigInt num1, num2, num3;
  char operation;

  ...

  return 0;
}
```

num1.size = 1;

while (isdigit(next_char)) {
    int i = 0;
    do {
        num1.digits[i] = toascii(next_char) - 48;
        i++;
    } while (isdigit(next_char));
    num1.digits[i] = toascii(next_char) - 48;
    (isdigit(next_char) ? cin.get(next_char) : do
        int i = 0;
        char next_char;
    }
} // somewhere inside getinput, needs copy
// reading in a large integer digit by digit