CS250 Intro to CS II

Spring 2013

Chapter 9 - Pointers
Reading: pp. 491-500

- Pointers are one of the most powerful features of C++

- Pointers give programmers more control over the computer's memory

- A pointer is the memory address of a variable

- A pointer is one of the most difficult and important concepts in C/C++

Pointer Declarations

- The memory address of a variable can be stored in another variable called a pointer

- Pointers are declared using the ***** operator

- The following declares a pointer to an integer
  - `int *pLength;`

- In the following statement, `length` is an integer and `pLength` is a pointer to an integer
  - `int *pLength, length;`

Pointer Problem

```cpp
#include <iostream>

using namespace std;

int main ()
{
  char *pCh, ch;
  cout << "Size of pCh is " << sizeof (pCh) << endl;
  cout << "Size of ch is " << sizeof (ch) << endl;

  return EXIT_SUCCESS;
}
```
What is the difference between pCh and ch?

What is the output from the above program?

Address Operator

- How do we assign the address of a variable to a pointer?

- Use the address operator (`&`)

- `&` returns the operand's memory address

- Example:
  - `pLength = &length;`

Address Operator

- Address operator cannot be applied to constants

  - `int *pX, x = 5;`

  - `const int NUM = 98;`

  - `pX = &x`   `// NO ERROR`

  - `pX = &NUM;`   `// ERROR`

  - `pX = &8;`   `// ERROR`

Pointer Operations

```
int x, *pX;
x = 8;    // set x to a value of 8
pX = &x; // set the pointer variable to point
          // to the address of x


cout << "x is: " << x << endl;
cout << "Size of x is: " << sizeof(x) << endl;
cout << "Address of x is: " << pX << endl;
cout << "Address of x is: " << &x << endl;
```

Indirection Operator

- How can we use the pointer variable to modify the value in the variable?

  - i.e. how to use `pX` to change the value of `x`

- *Answer:* use the indirection operator (`*`)

- The `*` operator <u>dereferences</u> the pointer

  - You are actually working with whatever the pointer is pointing to

- Using the example on the previous slide

  - `cout << "Value pX is pointing to is: " << *pX << endl;`

Indirection Operator

- Change the value of **x** from 8 to 10 using the pointer variable?

- Change the value of **x** to a value entered by the user using the indirection operator?

Question

Exactly what happens when the following program is compiled and executed?

```cpp
#include <iostream>

using namespace std;

int main ()
{
  int x, *pX;
  x = 8;
  *pX = 2;
  cout << "x = " << x << "*pX = " << *pX << endl;

  return EXIT_SUCCESS;
}
```

this Pointer

- functions - only one copy of each function exists in memory independent of the number of objects instantiated using the class declaration

- data members - each unique object of a particular class has space allocated for the data members of the class

- this - is a special built-in pointer available to a class's member functions. this points to the instance of the class making the function call

- this is passed as a hidden argument to all nonstatic member functions

RationalSet

- ## What do we return?

```
27  RationalSet RationalSet::add (const Rational &cRational)
28  {
29     if (!isInSet (cRational))
30     {
31        mcRational[mNumRationals] = cRational;
32        ++mNumRationals;
33     }
34
35     return
36  }
```

Rational Interface

```cpp
#ifndef RATIONAL_H
#define RATIONAL_H

using namespace std;

class Rational
{
public:
  Rational(int, int);
  print();

private:
  int numerator;
  int denominator;
};

#endif
```

Rational Implementation

```cpp
#include "Rational.h"

Rational::Rational(int numerator, int denominator)
{
   (*this).numerator = numerator;
   (*this).denominator = denominator;
}


Rational::print()
{
   cout << numerator << '/' << denominator;
}
```

Accessing data members

- Accessing data members using pointers

- `(*this).numerator` can be replaced with

- `this->numerator`