Overview of Pointers, Virtual Functions, Abstract classes

"Objects pointers, Abstract classes,Virtual functions, Friend functions and classes, Static functions"

Fundamentals of OOPs

Shakirullah Waseeb
shakir.waseeb@gmail.com

Nangarhar University

November 8, 2017

Agenda

1. Pointer Overview
   - Overview
   - Basic Concepts
   - Pointers and Functions, Pointers and Strings
   - The new and delete Operators
   - Pointers to Objects
2. Questions and Discussion

Agenda

Introduction

- **Pointers:** to access memory by its address rather than by its name, some common uses are
  - accessing array elements
  - modifying the original argument passed to function
  - obtaining memory from the system
  - creating data structures such as linked list

Agenda

Basic Concepts

pqqq

- The **address-of** Operator &; to get the address

```
int main (){
  int  somevar;
   cout « &somevar;
}
```

- The Pointer Variables

```
int main (){
  int  somevar;
  int*  ptr;
   ptr = &somevar;
   cout « ptr;
}
```


Basic Concepts –continue

- Accessing the variable pointed to

```
int  somevar;
int*  ptr;
 ptr = &somevar;
 cout « *ptr;
```

- Pointer to void

```
int  someint = 5;
float  somefloat = 9.3;
void*  ptr;
 ptr = &someint;
 cout « ptr « " : "«*reinterpret_cast<int*>(ptr);
 ptr = &somefloat;
 cout « ptr « " : "«*reinterpret_cast<float*>(ptr);
```


Agenda

Pointers and Functions

- We can pass arguments to a function as:
  - By value

```
void somefun(int param){}
int x = 8;
somefun(x);
```

  - By reference

```
void somefun(int& param){}
int x = 8;
somefun(x);
```

  - By passing pointers

```
void somefun(int* param){}
int x = 8;
somefun(&x);
```


Pointers and Strings

- As strings are array of characters therefore we ca interpret them as pointers as below :

```
char str1[] = "Some constant string";
char* str2 = "Some constant pointer string";
 cout « str1 « endl; //Some constant string
 cout « str2 « endl; //Some constant pointer string
 cout « ++str2 « endl; //ome constant pointer string
```


Agenda

The new and delete Operators

- **new** Operator: a different approach to obtain blocks of memory
- It obtains memory from operating system and returns a pointer to its starting point
- We must release the memory assigned by **new** operator
- Thus use the **delete** operator to release the memory

Example

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

int main {
      char* str = "some string";
      int size = strlen(str);
      char* str2ptr;
      str2ptr = new char[size+2];
      strcpy(str2ptr,str);
      cout «"str2ptr : "«str2ptr«endl;
}
```

Agenda

Pointers to Objects

- Pointers can point simple data types and arrays but can also point to objects
- Sometimes, however it is not known how many objects of a given class to create as hardcoded
- In such cases, we can use **new** to create objects during program execution

Example

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

int main {
      Counter* cntr;
      cntr = new Counter;
      cntr -> getCount();
}
```


Your Turn: Time to hear from you!

1

[1] https://fensafitters.files.wordpress.com/2013/07/3d095.jpg

References

Robert Lafore

Object-Oriented Programming in C++, 4th Edition .
2002.

Piyush Kumar

Object oriented Programming (Using C++)
http://www.compgeom.com/ piyush/teach/3330