Constructors and Destructors

Last updated 3/10/19

Constructors and Destructors

- Constructor
  - Member function automatically called when an object is created
  - Can do anything a normal function can do
  - Typically used to initialize an object's member variables

  - Must be public
  - Must have the same name as the Class
  - No return type

  - If not provided – automatically created by the compiler

Constructors and Destructors

- Default Constructor
  - No parameters are passed when the object is created
  - Should always be present

```
className(void);               // declaration

className::className(void){    // definition

className my_object;           // object creation
```

Note: no return type

Constructors and Destructors

- Default Constructor

```
/*
 * Cons.h
 *
 *  Created on: Feb 27, 2019
 *      Author: johnsontimoj
 */
//
// Constructor class example Class
// Used to show constructor operation
//
#ifndef CONS_H_
#define CONS_H_

// Cons class declaration

class Cons{
  // member data
  private:
    double var1;
  // member functions
  public:
    Cons(void);
    void setVar1(double v1);
    double getVar1(void);
};

#endif /* CONS_H_ */
```

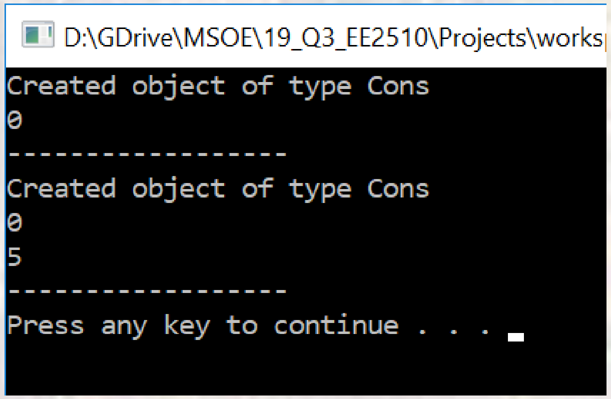
```
/*
 * Cons.cpp
 *
 *  Created on: Feb 27, 2019
 *      Author: johnsontimoj
 */
//
// Constructor class example Class
// Used to show constructor operation
//

// Implementation of Cons class

#include "Cons.h"
#include <iostream>
#include <cstdlib>
using namespace std;

//////////////////////////////////////////////////////
// Default Constructor
/////////////////////////////////////////////////////
Cons::Cons(void){
  var1 = 0.0;
  cout << "Created object of type Cons\n";
}
//////////////////////////////////////////////////////
// setVar1 - sets the var1 variable
/////////////////////////////////////////////////////
void Cons::setVar1(double v1){
  var1 = v1;
}

//////////////////////////////////////////////////////
// getVar1 - gets the var1 member variable
/////////////////////////////////////////////////////
double Cons::getVar1(){
  return var1;
}
```

Normally we would not do
I/O inside a class function
This is for educational purposes

Constructors and Destructors

- Default Constructor

```cpp
/*
 * cons_ex1.cpp
 *
 *  Created on: Feb 27, 2019
 *      Author: johnsontimoj
 */

/////////////////////////////////////////////
// program to test the Cons class
// and demonstrate default constructors
////////////////////////////////////////////

#include <iostream>
#include "Cons.h"
using namespace std;

int main(void){
  double foo;

  /////////////
  // Create a few objects
  /////////////
  Cons obj1;
  foo = obj1.getVar1();
  cout << foo << endl;
  cout << "------------------\n";

  Cons obj2;
  cout << obj2.getVar1() << endl;
  obj2.setVar1(5);
  cout << obj2.getVar1() << endl;
  cout << "------------------\n";

  system("pause");
  return 0;
}
```



D:\GDrive\MSOE\19_Q3_EE2510\Projects\works

```
Created object of type Cons
0
------------------
Created object of type Cons
0
5
------------------
Press any key to continue . . .
```

Constructors and Destructors

- Generalized Constructor
  - Parameters can be passed to the constructor when the object is created

```
className(formal param list);                  // declaration

className::className(formal param list){       // definition

className my_object(actual param list);        // object creation
```

Note: no return type

Constructors and D

- Generalized Constructor

```
/*
 * Cons2.h
 *
 *  Created on: Feb 27, 2019
 *      Author: johnsontimoj
 */
//
// Constructor class example Cons2
// Used to show Constructor operation
//
#ifndef Cons2_H_
#define Cons2_H_

// Cons2 class declaration

class Cons2{
  // member data
  private:
    double var1;
  // member functions
  public:
    Cons2(double v1);      // constructor
    Cons2(void);           // default constructor
    void setVar1(double v1);
    double getVar1(void);
};

#endif /* Cons2_H_ */
```

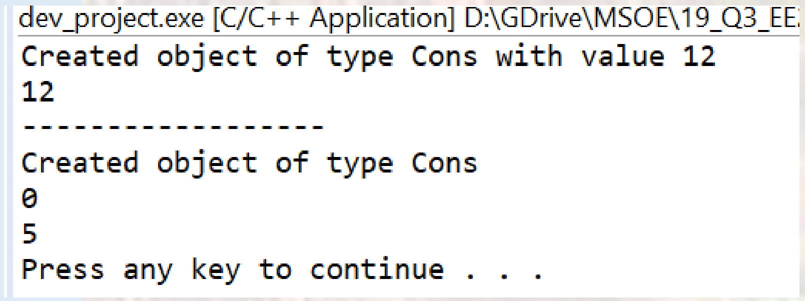
```
/*
 * Cons2.cpp
 *
 *  Created on: Feb 27, 2019
 *      Author: johnsontimoj
 */
//
// Constructor class example Cons2
// Used to show constructor operation
//

// Implementation of Cons2 class

#include "Cons2.h"
#include <iostream>
#include <cstdlib>
using namespace std;

////////////////////////////////////////////////////
//  Constructor
///////////////////////////////////////////////////
Cons2::Cons2(double v1){
  var1 = v1;
  cout << "Created object of type Cons with value " << var1 << "\n";
}

////////////////////////////////////////////////////
// Default Constructor
///////////////////////////////////////////////////
Cons2::Cons2(void){
  var1 = 0.0;
  cout << "Created object of type Cons\n";
}
////////////////////////////////////////////////////
// setVar1 - sets the var1 variable
///////////////////////////////////////////////////
void Cons2::setVar1(double v1){
  var1 = v1;
}

////////////////////////////////////////////////////
// getVar1 - gets the var1 member variable
///////////////////////////////////////////////////
double Cons2::getVar1(){
  return var1;
}
```

Normally we would not do
I/O inside a class function
This is for educational purposes

Constructors and Destructors

- Generalized Constructor

```
/*
 * cons_ex2.cpp
 *
 *  Created on: Feb 27, 2019
 *      Author: johnsontimoj
 */

/////////////////////////////////////////////
// program to test the Cons2 class
// and demonstrate generalized constructors
////////////////////////////////////////////

#include <iostream>
#include "Cons2.h"
using namespace std;

int main(void){
  double foo;

  /////////////
  // Create a few objects
  /////////////
  Cons2 obj1(12);
  foo = obj1.getVar1();
  cout << foo << endl;
  cout << "------------------\n";

  Cons2 obj2;
  cout << obj2.getVar1() << endl;
  obj2.setVar1(5);
  cout << obj2.getVar1() << endl;
  cout << "------------------\n";

  system("pause");
  return 0;
}
```


dev_project.exe [C/C++ Application] D:\GDrive\MSOE\19_Q3_EE

```
Created object of type Cons with value 12
12
------------------
Created object of type Cons
0
5
Press any key to continue . . .
```

Constructors and Destructors

- Destructor
  - Member function automatically called when an object is destroyed
    - When the function that created the object returns
    - When the program completes
  - Can do anything a normal function can do
  - Typically used to clean up an object's dynamic memory
  - Must be public
  - Must have the same name as the Class preceded by ~
  - No return type
  - If not provided – automatically created by the compiler

Constructors and Destructors

- Destructor
  - No parameters are passed
  - Should always be present

```
~className(void);                    // declaration

~className::className(void){ // definition
```

Note: no return type

Constructors and

- Destructor

```
/*
 * Des.h
 *
 *  Created on: Feb 27, 2019
 *      Author: johnsontimoj
 */
//
// Destructor class example Des
// Used to show destructor operation
//
#ifndef DES_H_
#define DES_H_

// Des class declaration

class Des{
  // member data
  private:
    double var1;
  // member functions
  public:
    Des(void);// constructor
    ~Des(void);// destructor
    void setVar1(double v1);
    double getVar1(void);
};

#endif /* DES_H_ */
```

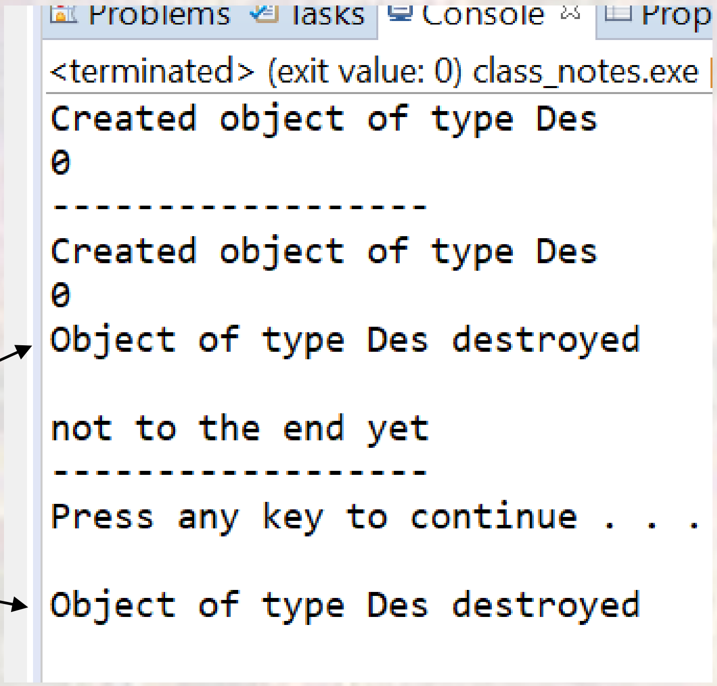
```
/*
 * Des.cpp
 *
 *  Created on: Feb 27, 2019
 *      Author: johnsontimoj
 */
//
// Destructor class example Des
// Used to show destructor operation
//

// Implementation of Des class

#include "Des.h"
#include <iostream>
#include <cstdlib>
using namespace std;

////////////////////////////////////////////////////
// Default Constructor
///////////////////////////////////////////////////
Des::Des(void){
  var1 = 0.0;
  cout << "Created object of type Des\n";
}

////////////////////////////////////////////////////
// Default Destructor
///////////////////////////////////////////////////
Des::~Des(void){
  // nothing to do here
  cout << "Object of type Des destroyed\n";
}

////////////////////////////////////////////////////
// setVar1 - sets the var1 variable
///////////////////////////////////////////////////
void Des::setVar1(double v1){
  var1 = v1;
}

////////////////////////////////////////////////////
// getVar1 - gets the var1 member variable
///////////////////////////////////////////////////
double Des::getVar1(){
  return var1;
}
```

Normally we would not do
I/O inside a class function
This is for educational purposes

Constructors and Destructors

•

```
/*
 * des_ex1.cpp
 *
 *  Created on: Feb 27, 2019
 *      Author: johnsontimoj
 */

/////////////////////////////////////////////
// program to test the Des class
// and demonstrate default destructors
////////////////////////////////////////////

#include <iostream>
#include "Des.h"
using namespace std;

int main(void){
  double foo;

  /////////////
  // Create a few objects
  /////////////
  Des obj1;
  foo = obj1.getVar1();
  cout << foo << endl;
  cout << "------------------\n";

  for(int i=0; i<1; i++){
    Des obj2;
    cout << obj2.getVar1() << endl;
  }

  cout << "\nnot to the end yet" << endl;
  cout << "------------------\n";

  system("pause");
  return 0;
}
```



```
<terminated> (exit value: 0) class_notes.exe
Created object of type Des
0
------------------
Created object of type Des
0
Object of type Des destroyed

not to the end yet
------------------
Press any key to continue . . .

Object of type Des destroyed
```