Sorting Circles

Sorting Circles

- Let's read a collection of Circles from a file and sort them by size.

Sorting Circles

- Let's read a collection of Circles from a file and sort them by size.

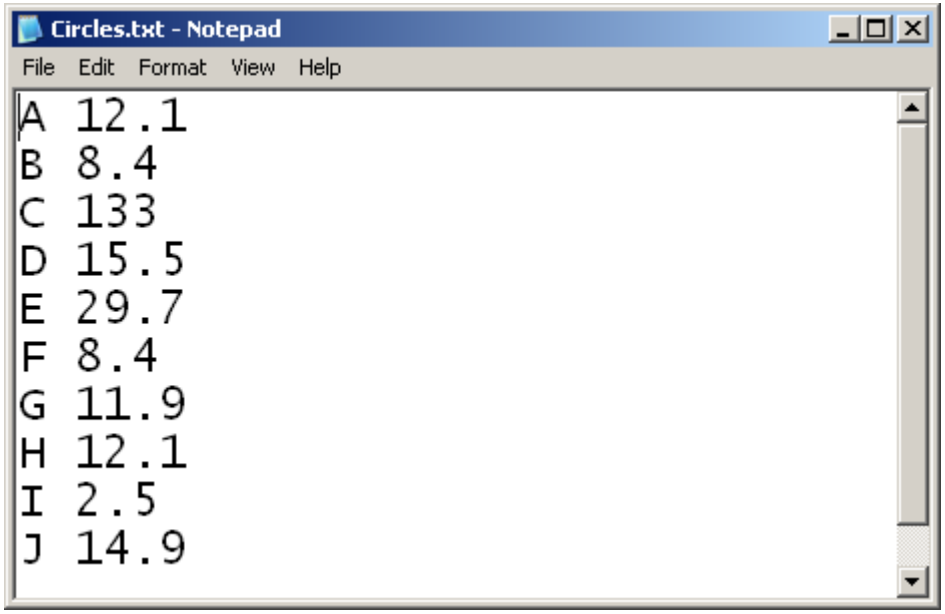
- Create new C++ console project.
  - Circle_Sort

- Download source files from last class
- http://www.cse.usf.edu/~turnerr/Object_Oriented_Design/Downloads/Circle_Demo3/
  - Add Circle.h, Circle.cpp, and main.cpp to new project

The Circles

- Download data file Circles.txt from the Downloads area to your project folder.

http://www.csee.usf.edu/~turnerr/Object_Oriented_Design/Downloads/2016_01_29/



main()

Add at the top:

```
#include <fstream>
```

main()

```
int main()
{
    Circle circles[20];
    cout << "This is Circle Sort\n";
    int count = -1;

    count = Get_Circles(circles, 20);
    if (count < 1)
    {
        cout << "Failed to read Circles file\n";
        cin.get();
        return -1;
    }

    cout << count << " Circles read\n";
    Display_Circles(circles, count);

   cout << "Normal completion\n";
    cin.get();
}
```

Get_Circles()

```
// Read circle data from a text file, set up Circle objects
// in caller's array, and return number of Circles.
// Negative count indicates error.
int Get_Circles(Circle* Circle_Array, int Max_Circles)
{
    int count = 0;
    ifstream infile;
    string name;
    double radius;
    infile.open("circles.txt");

    if (!infile.is_open())
    {
        cout << "Failed to open file\n";
        return -1;     // Error
    }
```

Get_Circles()

```
// Input file is open
while ((count < Max_Circles) && infile.good())
{
    string name;
    double radius;
    infile >> name;
    infile >> radius;

    if (!infile.good())
    {
        break;
    }
    Circle c(name, radius);
    Circle_Array[count++] = c;
}
```

Get_Circles()

```
    if (infile.eof())
    {
        cout << endl << "End of file \n";
    }
    else
    {
        cout << endl  << "Error reading file\n";
        count = -1;
    }

    infile.close();
    return count;
}
```

Display_Circles()

```
void Display_Circles(const Circle* circles, int nr_circles)
{
    cout << endl;
    for (int i = 0; i < nr_circles; ++i)
    {
        circles[i].Display();
    }
    cout << endl;
}
```

Build

Error!

Error List

Entire Solution | 2 Errors | 1 Warning | 0 Messages

Search Error List

	Code	Description	Project	File	Line	Supp...
abc		no default constructor exists for class "Circle"	Circle_Demo	Circle_Test.cpp	109	
⚠	C4101	'radius': unreferenced local variable	Circle_Demo	circle_test.cpp	58	
✖	C2512	'Circle': no appropriate default constructor available	Circle_Demo	circle_test.cpp	109	

- For an array of objects, the class must provide a default constructor.
  - Used to initialize the objects in the array.
- Add to Circle.h: `Circle() {};`

Program in Action

c:\Documents and Settings\Rollins\Desktop\Circle_Demo3\Debug\Circle_Demo3.exe

```
This is Circle Sort

End of file
10 Circles read
A is a Circle of radius 12.1
B is a Circle of radius 8.4
C is a Circle of radius 133
D is a Circle of radius 15.5
E is a Circle of radius 29.7
F is a Circle of radius 8.4
G is a Circle of radius 11.9
H is a Circle of radius 12.1
I is a Circle of radius 2.5
J is a Circle of radius 14.9
```

Sort Circles

```
void Sort_Circles(Circle* circles, int nr_circles)
{
    bool swap_done = false;
    do
    {
        swap_done = false;
        for (int i = 0; i < nr_circles-1; ++i)
        {
            if (circles[i] > circles[i+1])
            {
                swap_circles(circles[i], circles[i+1]);
                swap_done = true;
            }
        }
    } while (swap_done);
}
```

Overloaded
> operator

Swap_Circles()

```
void swap_circles(Circle& c1, Circle& c2)
{
    Circle temp = c1;
    c1 = c2;
    c2 = temp;
}
```

Add to main()

```
    Sort_Circles(circles, count);

    cout << "Sorted circles:\n";
    Display_Circles(circles, count);

    cout << "Normal completion\n";
    cin.get();
}
```

Sorting Circles

c:\Documents and Settings\Rollins\Desktop\Circle_Demo3\Debug\Circle_Demo3.exe

```
This is Circle Sort

End of file
10 Circles read

A is a Circle of radius 12.1
B is a Circle of radius 8.4
C is a Circle of radius 133
D is a Circle of radius 15.5
E is a Circle of radius 29.7
F is a Circle of radius 8.4
G is a Circle of radius 11.9
H is a Circle of radius 12.1
I is a Circle of radius 2.5
J is a Circle of radius 14.9

Sorted circles:

I is a Circle of radius 2.5
B is a Circle of radius 8.4
F is a Circle of radius 8.4
G is a Circle of radius 11.9
A is a Circle of radius 12.1
H is a Circle of radius 12.1
J is a Circle of radius 14.9
D is a Circle of radius 15.5
E is a Circle of radius 29.7
C is a Circle of radius 133

Normal completion
```

A Better Solution

- Sorting the array of Circles directly requires moving entire Circle objects around.

- It is generally more efficient to sort an array of pointers.
  - Leave the Circle objects in place.

Modify main()

```
int main()
{
    Circle* circles[20];
    cout << "This is Circle Sort\n";
    ...
```

Modify Get_Circles()

```
// Read circle data from a text file, set up pointers to Circle objects
// in caller's array, and return number of Circles.
int Get_Circles(Circle** Circle_Array, int Max_Circles)
{
    int count = 0;
    ifstream infile;
    infile.open("circles.txt");

    if (!infile.is_open())
    {
        cout << "Failed to open file\n";
        return -1;        // Error
    }
```

Modify Get_Circles()

```
// Input file is open
while ((count < Max_Circles) && infile.good())
{
    char name[500];
    double radius;
    infile >> name;
    infile >> radius;

    if (!infile.good())
    {
        break;
    }
    Circle_Array[count++] = new Circle(name, radius);
}
...
```

Modify Display_Circles()

```
// Remove const from first parameter
void Display_Circles(Circle** circles, int nr_circles)
{
    cout << endl;
    for (int i = 0; i < nr_circles; ++i)
    {
        circles[i]->Display();
    }
    cout << endl;
}
```

Modify Sort_Circles()

```
void Sort_Circles(Circle** circles, int nr_circles)
{
    bool swap_done = false;
    do
    {
        swap_done = false;
        for (int i = 0; i < nr_circles-1; ++i)
        {
            if (*circles[i] > *circles[i+1])
            {
                swap_circles(circles[i], circles[i+1]);
                swap_done = true;
            }
        }
    } while (swap_done);
}
```

Modify Swap_Circles()

```
void swap_circles(Circle*& c1, Circle*& c2)
{
    Circle* temp = c1;
    c1 = c2;
    c2 = temp;
}
```

Works the Same

c:\Documents and Settings\Rollins\Desktop\Circle_Demo3\Debug\Circle_Demo3.exe

```
This is Circle Sort

End of file
10 Circles read

A is a Circle of radius 12.1
B is a Circle of radius 8.4
C is a Circle of radius 133
D is a Circle of radius 15.5
E is a Circle of radius 29.7
F is a Circle of radius 8.4
G is a Circle of radius 11.9
H is a Circle of radius 12.1
I is a Circle of radius 2.5
J is a Circle of radius 14.9

Sorted circles:

I is a Circle of radius 2.5
B is a Circle of radius 8.4
F is a Circle of radius 8.4
G is a Circle of radius 11.9
A is a Circle of radius 12.1
H is a Circle of radius 12.1
J is a Circle of radius 14.9
D is a Circle of radius 15.5
E is a Circle of radius 29.7
C is a Circle of radius 133

Normal completion
```