Lecture 4 Notes

Functions

If you find yourself repeating the same lines of code in many different parts of your program, you can use a function instead.

A function holds a sequence of statements. When given a name, a function can be used anywhere in your program by referring to it's name.

Listing 1: A program that uses a function

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

void hr()
{
   int i;
   for (i = 0; i < 78; ++i)
      cout << '-';
   cout << '\n';
}

int main()
{
   hr();
   cout << "Fancy"s << endl;
   hr();
}
```

In this example, the function `hr` displays a horizontal line. The statement `hr();` executes the function.

Parameters

Functions may accept input values via parameters.

Listing 2: Function hr with one parameter

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

void hr(char c)
{
   int i;
   for (i = 0; i < 78; ++i)
      cout << c;
   cout << '\n';
}

int main()
```

```
{
   hr('#');
   cout << "Fancy"s << endl;
   hr('#');
}
```

Now `hr` can output a horizontal line using any fill character. We can also add a second parameter for controlling the length of the line:

Listing 3: Function hr with two parameters

```
void hr(char c, int len)
{
   int i;
   for (i = 0; i < len; ++i)
      cout << c;
   cout << '\n';
}
```

Here are some ways we can invoke our new two parameter function:

```
hr('=', 32);   /* Prints a short line of equal signs. */
hr(':', 78);   /* Prints a long line of colons */
hr('_', 52);   /* Prints a medium line of underscores */
```

Return values

In the previous examples, the `hr` function used `void` in place of the return type; this means that our function does not return a value. We can rewrite our function to return a horizontal line in a string instead of directly writing to `cout`. We can also change the first parameter to use a string instead of just a single character, this will allow `hr` to create pretty line patterns:

Listing 4: Function hr with a return value

```
string hr(string s, int len)
{
   string t;
   int i;
   for (i = 0; i < len; ++i)
      t += s;
   t += '\n';
   return t;
}
```

Now we can use our `hr` function like this:

```
cout << hr("-="s, 39);
cout << "Fancy"s endl;
cout << hr("-="s, 39);
```

The disadvantage of all this is that it's more complicated to use `hr` now. What started off as a simple function to display a horizontal line has now become a two-parameter, all-purpose function that repeatedly concatenates a string into a new string.

We can have a new function called `strrep` (which stands for string repeat), and have an old version of `hr` use our new function:

```
string strrep(string s, int len)
{
   string t;
   int i;
   for (i = 0; i < len; ++i)
      t += s;
   return t;
}

void hr()
{
   cout << strrep(""s + '-', 78) << endl;
}
```

Now we have two functions: a simple to use `hr` function that prints a horizontal line of dashes, and a more general-purpose `strrep` for repeating a string.

The C++ Standard Library

The C++ standard library is huge, some of it's features are complicated to use. This section outlines a few simple functions provided by the standard library. You must include the appropriate header file before using these functions.

Functions from <string>

```
int stoi(string s);
```

`stoi` converts a string into an integer.

```
double stod(string s);
```

`stod` converts a string into a double.

Functions from <cstdlib>

```
exit();
```

`exit` terminates the program.

```
int abs(int n);
```

`abs` returns the absolute value of an integer.

```
void srand(int seed);
```

Initializes the built-in pseudo-random number generator. The value of seed can be anything you want.

```
int rand();
```

Returns a pseudo-random integer value greater than or equal to zero.

Functions from <ctime>

```
int time(int unused);
```

Returns the current time in seconds. For simplistic purposes, use the value zero as the first argument to this function like this: `time(0)`.

Functions from <cmath>

```
double fabs(double n);
```

`fabs` returns the absolute value of a real number.

```
double exp(double n);
```

`exp` returns the mathmatical constant e raised to the nth power.

```
double log(double n);
```

`log` returns the natural logarithm (base e) of n.

```
double log10(double n);
```

`log10` returns the common logarithm (base 10) of n.

```
double pow(double base, double exponent);
```

`pow` returns the value base raised to the power of the value exponent.

```
double sqrt(double n);
```

`sqrt` returns the square root of n.

```
double sin(double n);
double cos(double n);
double tan(double n);
```

These are the trigonometric functions.

```
double asin(double n);
double acos(double n);
double atan(double n);
```

These are the inverse trigonometric functions.

```
double round(double n);
```

`round` returns n rounded to the nearest whole number.