C++ Support Classes

Introduction

Dr. Johnson

School of Mathematics

2013
OUTLINE

1 REVIEW

2 FLOW CONTROL
   • Conditional flow
   • loops

3 FUNCTIONS AND POINTERS
   • Functions
   • Pointers

4 SUMMARY
OUTLINE

1 REVIEW

2 FLOW CONTROL
   • Conditional flow
   • loops

3 FUNCTIONS AND POINTERS
   • Functions
   • Pointers

4 SUMMARY
OUTLINE

1 Review

2 Flow Control
   - Conditional flow
   - loops

3 Functions and Pointers
   - Functions
   - Pointers

4 Summary
Topics:
- Computers and Programs;
- Syntax of C++;
- Data and Variables;
- Input and Output.

Aims:
- Understand the idea of programming a computer;
- Write a simple program to input and output data.
Today...

Topics:
- Flow control - if, else, for, do
- Functions - how and where to use them
- Pointers - dynamic allocation, pass by reference

Aims:
- Use functions, loops and if statement to control a program.
- Understand the concept of a pointer.
If, Else If and Else

- We can use if, else if, and else to control flow through the program.

```cpp
int i;
cout << " Enter a number " << endl;
cin >> i;
if(i<0)cout << " i is negative" << endl;
else if(i==0)cout << " i is zero" << endl;
else cout << " i is positive" << endl;
```
To execute more than one command on an if condition use blocks

```c
if(condition){
    // lots of commands in here
}
else {
    // and in here too.
}
```
FOR LOOPS

- The general form for a loop is

```c
for(initialisation; condition; increment) statement;
```

- We can loop over multiple commands using a block

```c
for(int i=0; i<10; i++){
    temp = i*10;
    cout << " value " << temp << endl;
}
```
FOR LOOPS

The general form for a loop is

\[
\text{for} (\text{initialisation}; \text{ condition}; \text{ increment}) \text{ statement;}
\]

We can loop over multiple commands using a block

\[
\text{for} (\text{int } i=0; i<10; i++)\{
\text{temp} = i*10;
\text{cout} << " value " << \text{temp} << \text{endl;}
\}
\]
The command `break` can be used to exit a loop.

```c
for(int loop=0; loop<iter_max; loop++)
{
    solve_for_U(u, y, U);
    if(residual(x, y, U)<tolerance) break;
}
```
Another alternative loop iteration is the while loop.

Their functional form is:

\[ \text{while}(\text{condition}) \text{statement}; \]

The statement is executed until the condition is true.
The condition is evaluated **before** the statement.

```cpp
int i=0;
while(i<100)
{
    i++;
    std::cout << " i= " << i << std::endl;
}
```
Another alternative loop iteration is the do–while loop. Their functional form is:

```
do {statement;}
while(condition);
```

The statement is executed until the condition is true. The condition is evaluated after the statement.

```
do {
solve_for_U(u,y,U);
}
while(residual(x,y,U)<tolerance);
```
The general syntax for a function is:

```
data type function_name(arguments)
{ function statements }
```

- Functions must be declared before the main program.
- All functions must return a value of the data type specified in the declaration.
- Even if this is `void`!
#include<iostream>
using namespace std

// square an integer
int square(int i)
{
    return i*i;
}

// Main Program
main()
{
    int number=5
    cout << square(number) << endl;
}
ACCESSING THE MATH LIBRARY

- Simply include the library at the top of your code:

```cpp
#include<cmath>
```

- All of the trigonometric, hyperbolic and exponential functions are present.
- There is also a `pow(x, y)` to raise `x` to the power `y`.
- and a `sqrt()` function.
Prototyping

- A function must be defined before it can be called.
- Use prototypes to declare functions before they are used.

```
data type function_name(arguments)
```

- The main body of the function can be placed somewhere else in the code (or even a separate file)
What’s in a Name?

- Pointers are an important mechanism in any computer program.
- In many languages pointers are hidden from the user.
- Pointers store the location of a value, rather than the value.

<table>
<thead>
<tr>
<th>Memory</th>
<th>Compiler</th>
</tr>
</thead>
<tbody>
<tr>
<td>0xffff8</td>
<td>0xffff4</td>
</tr>
<tr>
<td>6</td>
<td>0xffff8</td>
</tr>
<tr>
<td>a</td>
<td>ptr_a</td>
</tr>
<tr>
<td>0xffff8</td>
<td>0xffff4</td>
</tr>
</tbody>
</table>

Dr. Johnson

MATH60082
When we pass a variable into a function, the stored value is copied into the function, not the variable itself.

To change the value of the variable itself, we must pass a reference memory location of the variable into the function.

```c
void swap(double &a, double &b)
{
    NB. stuff in here
}
main()
{
    double a=1, b=2;
    swap(a, b);
    cout << " a " << a << " b " << b;
}
```
YOU DON’T NEED TO KNOW ANYTHING ABOUT POINTERS TO CODE MATHEMATICAL FUNCTIONS

JUST REMEMBER TO PUT & WHEN CHANGING VALUE INSIDE A FUNCTION!!!!
YOU DON’T NEED TO KNOW ANYTHING ABOUT POINTERS TO CODE MATHEMATICAL FUNCTIONS

JUST REMEMBER TO PUT & WHEN CHANGING VALUE INSIDE A FUNCTION!!!!
SUMMARY

Topics:
- Flow control - if, else, for, do
- Functions - how and where to use them
- Pointers - pass by reference

Aims:
- Use functions, loops and if statement to control a program.
- Use pass by reference.