The Basics of C++ (cont.)

闫宏飞

Institute of Network
Peking University

Oct 23, 2007
1. Introduction

2. Control Structures

3. Functions

4. Arrays
OBJECTIVES

- To use the `if` and `if...else` selection statements (ch4)
- To understand multiple selection using the `switch` selection statement (ch5)
- To use the `for/while/do...while` repetition statement to execute statements (ch4,5)
- To construct programs modularly from `functions` (ch6)
- To use the `array` data structure to represent a set of related data items (ch7)
Algorithms

- Before writing a program
  - Have a thorough understanding of problem
  - Carefully plan your approach for solving it

- Algorithms
  - The actions to execute
  - The order in which these actions execute
Control Structures

- **Sequence structure**
  - Programs executed sequentially by default

- **Selection structures**
  - if, if...else, switch

- **Repetition structures**
  - while, do...while, for
Single-entry/single-exit control statements

- Three types of control statements
  - Sequence statement
  - Selection statements
  - Repetition statements

- Combined in one of two ways
  - Control statement stacking
  - Connect exit point of one to entry point of the next
  - Control statement nesting
The Structure of an If Statement

```cpp
if ( TRUE ) {
    // Execute these statements if TRUE
}
else {
    // Execute these statements if FALSE
}
```
```cpp
#include <iostream>

using namespace std;

int main()
{
    int age;
    cout << "Please input your age: ";
    cin >> age;
    cin.ignore(); // Throw away enter

    if ( age < 100 ) { // If the age is less than 100
        cout << "You are pretty young! \n"; // show you it works...
    }

    else if ( age == 100 ) { // use else to show an example
        cout << "You are old \n";
    }

    else { // Executed if no other statement
        cout << "You are really old \n";
    }

    cin.get();
}
```
**Boolean operators OR, NOT, and AND**
- in a similar way to the comparison operators

**Examples**

A. `!( 1 || 0 )` ANSWER: 0

B. `!( 1 || 1 && 0 )` ANSWER: 0 (AND is evaluated before OR)

C. `!( ( 1 || 0 ) && 0 )` ANSWER: 1 (Parenthesis are useful)
Switch case statements are a substitute for long if statements that compare to an integral value.

switch (value) {
    case this:
        Code to execute if value == this
        break;
    case that:
        Code to execute if value == that
        break;
    ...
    default:
        Code to execute if value != this or that
        break;
}
#include <iostream>

using namespace std;

void studyChinese();
void studyEnglish();
void studyFrance();

int main()
{
    int input;

    cout << "1. Study Chinese
";
    cout << "2. Study English
";
    cout << "3. Study France
";
    cout << "4. Exit
";
    cout << "Selection: ";
    cin >> input;
A FOR Loop Program (cont.)

switch (input) {
    case 1:  // Note the colon, not a semicolon
        studyChinese();
        break;
    case 2:
        studyEnglish();
        break;
    case 3:
        studyFrance();
        break;
    case 4:
        cout << "Congratulation for studying! \n";
        break;
    default:
        cout << "Error, bad input, quitting \n";
        break;
}
cin.get();
FOR loops are the most useful type.

for (variable initialization; condition; variable update) {
    Code to execute while the condition is true
}


```cpp
#include <iostream>

using namespace std; // So the program can see cout and endl

int main()
{
    // The loop goes while x < 10, and x increases by one every loop
    for (int x = 0; x < 10; x++) {
        // Keep in mind that the loop condition checks
        // the conditional statement before it loops again.
        // consequently, when x equals 10 the loop breaks.
        // x is updated before the condition is checked.
        cout << x << endl;
    }
    cin.get();
}
```
WHILE loops are very simple.

while ( condition ) {
    Code to execute while the condition is true
}

```cpp
#include <iostream>

using namespace std; // So we can see cout and endl

int main()
{
    int x = 0; // Don't forget to declare variables
    while (x < 10) { // While x is less than 10
        cout << x << endl;
        x++; // Update x so the condition can be met eventually
    }
    cin.get();
}
```
DO...WHILE loops are useful for things that want to loop at least once.

```cpp
do {
} while ( condition );
```
#include <iostream>

using namespace std;

int main()
{
    int x;

    x = 0;
    do {
        // "Hello, world!" is printed at least one time
        // even though the condition is false
        cout << "Hello, world!\n";
    } while ( x != 0 );

    cin.get();
}
functions are blocks of code that perform a number of pre-defined commands to accomplish something productive.

- Functions will generally require a prototype.
- The prototype tells the compiler what the function will return,
- what the function will be called,
- as well as what arguments the function can be passed.

The general format for a prototype is simple:

```
return-type function_name(arg_type arg1, ..., arg_type argN);
```
```cpp
#include <iostream>
using namespace std;

int mult(int x, int y);

int main()
{
    int x, y;

    cout << "Please input two numbers to be multiplied: ";
    cin >> x >> y;
    cin.ignore();
    cout << "The product of your two numbers is " << mult(x, y) << "\n";
    cin.get();
}

int mult(int x, int y)
{
    return x * y;
}
```
Arrays

Arrays are useful critters that

- often show up when it would be convenient to have one name for a group of variables of the same type
- that can be accessed by a numerical index.

- Arrays are essentially a way to store many values under the same name
- You can make an array out of any data-type including structures and classes.
Syntax for Declaring an Array

One way to visualize an array is like this:

[ ] [ ] [ ] [ ] [ ] [ ]

declaring an array

```c++
int examplearray[100]; /* This declares an array */
char astring[100];
```
#include <iostream>

using namespace std;

int main()
{
    char astring[10];
    int i = 0;
    cin >> astring;
    for (i = 0; i < 10; ++i) {
        if (astring[i] == 'a') {
            cout << "You entered an a!
";
        }
    }
}