Chapter - 28
C++'s Dustier Corners

Less used features of C++

do/while

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

goto

```
  for (x = 0; x < X_LIMIT; ++x) {
       for (y = 0; y < Y_LIMIT; ++y) {
           if (data[x][y] == 0)
               goto found;
       }
    }
    std::cout << "Not found\n";
    exit(8);

found:
```

Why do we fail to print an error for bad commands?

```
#include <iostream>
#include <stdlib.h>


int main(){
    char  line[10];
    while (1) {
        std::cout << "Enter add(a), delete(d), quit(q): ";
        std::cin.getline(line, sizeof(line));

        switch (line.at(0)) {
        case 'a':          std::cout << "Add\n";    break;
        case 'd':          std::cout << "Delete\n"; break;
        case 'q':          std::cout << "Quit\n";   exit(0);
        defualt:
            std::cout << "Error:Bad cmd " << line[0] << '\n';
            break;
        }
    }
    return (0);
}
```

More features

The ?: Construct

```
(expression) ? expr1 : expr2
```

Example:

```
amount_owed = (balance < 0) ? 0 : balance;
```

The , Operator

```
if (total < 0) {
    std::cout << "You owe nothing\n";
    total = 0;
}
```

Is the same as:

```
if (total < 0)
    std::cout << "You owe nothing\n",total = 0;
```

Except the first version is clearer.

Overloading the () operator

```
class example {
    public:
        int operator () (int i) {
            return (i*2);
        }
};
// ....
     example example_var;


     j = example_var(3);
     // j is assigned the value 6 (3*2)
```

Pointers to members

```
class sample {
    public:
        int i;    // A couple of member variables
        int j;
};

int sample::* data_ptr;

data_ptr = &sample::i;

sample a_sample; // A typical sample
sample b_sample;

std::cout << a_sample.*data_ptr << '\n';
std::cout << b_sample.*data_ptr << '\n';

std::cout << a_sample.*data_ptr << '\n';

sample *sample_ptr = &sample1;

std::cout << sample_ptr->*data_ptr << '\n';
```