More on Basic C++ Programming

Hsuan-Tien Lin

Dept. of CSIE, NTU

March 5, 2013

What We Have Done (Chapter 1)

- pointers and references

What We Will Do

- references and parameter passing (Chapter 1)
- template and STL (Chapter 1)
- arrays and linked list (Chapter 3)

References

What happens in memory?

```
class Record{ char* name; int score; };

Record r1;
Record r2;
Record* pr1 = &r1;   pr1->name   (r1.name)
Record* pr2 = &r2;
Record& r3 = r2; //reference (needs to be initialized)
r2 = r1;
pr2 = pr1;           r3.name   r2.name
```

Pointer/Reference: Key Point

- pointer: the “address” to the instance;
  用"海角七號"就可以找到老太太
- reference: the “other name” of the instance;
  用"小島友子"也可以稱呼老太太
- the “original” variable: holds the reference;
  一張"身份證"，上面寫著(住海角七號)
- pointer variable: holds the pointer;
  一個"信封"，上面寫著海角七號
- reference variable: holds the reference;
  一張"名片"，上面寫著(住海角七號)

```
老人  老太太;
老人*  信封= &老太太; //存放住址
老人& 小島友子= 老太太; //用別名
      老太太.回憶();
      信封->回憶();
      小島友子.回憶();
```

Argument/Parameter

```
Record htlin;
htlin.score = 59;
change1(htlin, 100);
change2(&htlin, 100);
change3(htlin, 100);

void change1(Record r1, int score){
  r1.score = score;
}
void change2(Record* pr1, int score){
  pr1->score = score;
}
void change3(Record& r1, int score){
  r1.score = score;
}
```

Reference Argument/Parameter: Key Point

argument ⇒ parameter: by copying
(unless specifying reference), **same for return value**

template in C++

integer polynomials, double polynomials, fraction polynomials, ...

```
template <class T>
class poly{
T coeff[1000];
poly<T> operator+(poly<T>& p){
...
}
};

poly<double> pd;    poly<int> pi;
```

Standard Template Library (STL)

- e.g. `vector`: dynamically growing array (will discuss more soon)

```
#include <vector>
using namespace std;
vector<int> intarray;

intarray.resize(20); intarray[3] = 5;
```

vector<double>

vector<Complex>