Object-Oriented Programming

Using C++

Course Contents

- Introduction to object-oriented programming…
- …with a strong software engineering foundation…
- …aimed at producing and maintaining large, high-quality software systems.

Buzzwords

- encapsulation
- inheritance
- polymorphism
- overriding
- interface
- cohesion
- coupling
- collection classes
- template
- responsibility-driven design

Textbooks

- C++ Prime
- Thinking In C++,Ver. 2,Vol. 1 & 2
- C++ Primer,Ver 5
- References:
  - The C++ Programming Language
  - C++: The Core Language
  - Essential C++
  - Effective C++
  - Inside the C++ Object Model
  - C++ Templates

Bruce Eckel

- BRUCE ECKEL is the author of “Thinking in C++” , who won the Software Development Jolt Award for best book of 1995. He's been professionally programming for 20 years and has been teaching people throughout the world how to program with objects since 1986. He was a voting member of the C++ Standards Committee.
- http://mindview.net

Assessment

1. In-class quiz: 5%, on 学在浙大 or paper work
2. Assignments: 16%, one problem set for each week, on PTA, due next lecture
3. 8 Lab/Project: 24%
4. Mid-Term Exam: 5% on PTA, 90-min at lab period, week 9
5. Final Exam: 50%

Tools for C++

- TDM GCC64 (Windows) or
- clang+llvm (macOS)
- Visual Studio Code
- 学在浙大有安装视频
- 可以继续使用Dev C++

Introduction to C++

The trip begins...

学哪个语言

https://www.tiobe.com/tiobe-index/

其他语言?

- 几乎都是C-like语言
    - 现代的编程语言在语法上的差异很小
- 语言的能力/适用领域主要是由
    - 库, and
    - 传统所决定的

结论建议

- 对计算机本身(体系结构、操作系统、编译)感兴 趣—>C
- 想编程解决手头的问题(统计、AI、桌面小程序) —>Python
- 有明确的需求(求职)—>人家要什么学什么 (PHP、JavaScript、C++)
  - C++是写库的语言
- 还没想好 —>Java

Introduction to C++

The trip begins...

The C Language

- Strengths
  - Efficient programs
  - Direct access to machine, suitable for OS and ES
  - Flexible
- Weakness
  - Insufficient type checking
  - Poor support for programming-in-the-large
  - Procedure-oriented programming

Bjarne Stroustrup

- http://www.research.att.com/~bs/homepage.html
- C++ was first designed and implemented by Bjarne Stroustrup, AT&T, early 1980's
- Oct. 2002, Stroustrup visited Zhejiang Univ.
- The Design and Evolution of C++, Bjarne Stroustrup, Addison-Wesley, ISBN 0-201-54330-3

Brief history of C++ (1)

- 1978: BS at Cambridge, UK.
  - Simulation program in Simula
  - Supports classes, inheritance, and type check Poor performance

- 1979: BS at AT&T Labs, Cpre, C w/ classes
- 1980: most C++ features but virtual functions
- 1983: C++ w/ virtual functions, named C++ by Rick Mascitti 1985: Cfront
- 1985:“The C++ Programming Language”
- 1990:ANSI C++ Committee ISO/ANSI Standard C++ in 1998: ISO/IEC 14882 (http://www.open-std.org/jtc1/sc22/ wg21/)

Goal for C++

- Support for object oriented programming (from SmallTalk) to combine flexibility and efficiency of C

C and C++

- C++ builds on C
  - Knowledge of C helps you in C++
  - C++ support more styles of programming
  - C++ provides more features 26

C++ improvements

- Data abstraction
- Access control
- Initialization & cleanup
- References
- Function overloading
- Streams for I/O
- Name control
- Operator overloading
- More safe and powerful memory management
- Templates
- Exception handling

C++ C++ can be viewed as a “better” C

- `C++` => `C=C+1`
  but...
- C++ is not C
  - Focus on C++ as a language in its own right
- C++ is a hybrid language, supports
  - Procedure-oriented programming
  - Object-oriented programming
  - Generic programming

The First C++ Program

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

int main()
{
  cout << "Hello, World! I am " << 18 < Today!" << endl;
  return 0;
}
```

Read input

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

int main() {
  int number;
  cout << "Enter a decimal number: ";
  cin >> number;
  cout << "The number you entered is " << number <<" endl;
  return 0;
}
```

string

- `string` is a class in C++
  - You must add this at the beginning of your code `#include <string>`
- Define variable of string like other types: `string str;`
- Initialize it w/ string constant: `string str = "Hello";`
- Read and write string w/ `cin` / `cout` : `cin >> str; cout << str;`

Assignment for string

```
char charr1[20];
char charr2[20] = "jaguar";
string str1;
string str2 = "panther";
carr1 = char2; // illegal
str1 = str2; // legal
```

Concatenation for string

```
string str3;
str3 = str1 + str2;
str1 += str2;
str1 += "lalala";
```

length

```
s.length();
```

- The dot `.` is an operator that retrieve a member of a struct in C
- It is still that operator in C++ that retrieve a member of an object in C

create a string

```
string(const char *cp, int len);
string(const string& s2, int pos);
string(const string& s2, int pos, int len);
```

sub-string

```
substr(int pos, int len);
```

alter string

```
assign();
insert(const string&, int len); insert(int pos, const string& s); erase();
append();
replace();
```

search string

```
find()
```

Pointers to Objects

```
string s = "hello";
string* ps = &s;
```

Operators with Pointers

- `&` : get address
- `ps = &s;`
- `*` : get the object
  - `(*ps).length()`
- `->` : call the function
  - `ps->length()`

Two Ways to Access

- `string s;`
  - `s` is the object itself
- `string *ps;`
  - `ps` is a pointer to an object

Object vs Pointer

- `string s;`
  - At this line, object `s` is created and initialized
- `string *ps;`
  - At this line, the object `ps` points to is not known yet

Assignment

```
string s1, s2;
s1 = s2;
string *ps1, *ps2;
ps1 = ps2;
```

dynamically allocated memory

- `new`
  - `new int;`
  - `new Stash;`
  - `new int[10]`
- `delete`
  - `delete p;`
  - `delete[] p;`
- new is the way to allocate memory as a program runs. Pointers become the only access to that memory
- delete enables you to return memory to the memory pool when you are finished with it.

Dynamic Arrays

```
int * psome = new int [10];
delete[] psome;
```

- The `new` operator returns the address of the first element of the block.
- The presence of the brackets tells the program that it should free the whole array, not just the element

The new-delete mech.

```
int *p=new int;
int *a=new int[10];
16 160
Student *q=new Student();
Student *r=new Student[10];
delete p;
a++;delete[] a;
delete q;
delete r;
delete[] r;
```

Tips for new and delete

- Don't use `delete` to free memory that `new` didn't allocate.
- Don't use `delete` to free the same block of memory twice in succession.
- Use `delete []` if `new []` was used to allocate an array.
- Use `delete` (no brackets) if `new` was used to allocate a single entity.
- It's safe to apply `delete` to the null pointer (nothing happens).

What we've learned today?

- A brief history of C++
- Input and output in C++ with `cin` and `cout`
- The `string` class, the dot `.` operator
- Pointer to an object, the arrow `->` operator
- Dynamic memory allocation with `new` and `delete`