From C to C++

Hsuan-Tien Lin

Dept. of CSIE, NTU

March 3, 2020

Key Differences of C++ That You May Need

What is C++?

“evolving” extension of C

Reference versus Pointer

```
c_swap(&x, &y);

void c_swap(int* pa, int* pb){
 int tmp;
 tmp = (*pa);
 (*pa) = (*pb);
 (*pb) = tmp;
}

cpp_swap(x, y);
void cpp_swap(int& a, int& b){
 int tmp;
 tmp = a;
 a = b;
 b = tmp;
}
```

reference: C++ way of smarter/safer “pointer”

Class versus Structure

```
typedef struct
{ double real;double img; } complex;

complex multiply(complex a, complex b){
   /* create a complex variable res */
   /* put a.real * b.real - a.img * b.img in res.real */
   ....
}

class complex{
public:
  double real;
  double img;

  void multiplyby(complex another){...}
}
```

class: C++ way of structure “with actions”

Operator versus Function

```
C:
int a, b, c;
c = a + b;  /*"built-in plus of int" */

complex A, B, C;
C = plus(A, B);

C++:
complex A, B, C;
C = A + B;
//C++ compiler "translate"
C = operator+(A, B);
C = A.operator+(B);
```

operator overloading: C++ way of programming with (some) complicated classes more easily

Template versus Copy/Paste

```
int int_arr[10];
double double_arr[10];
complex complex_arr[10];

/*write this first */
void int_sort(int int_arr){ .... }

/*copy/paste/replace*/
void double_sort(double double_arr){ ... }

template <class T>
void sort(T arr){ ... }

int int_arr[10];
sort(int_arr);
sort(complex_arr);
```

template: C++ way of automating safer copy/pasting by compilers

Standard Template Library in C++

STL

one data structure/algorithm can be applied to many classes