«C++ for C Programmer» Notes

by Firmin Martin

Week 1

From C to C++

C	C++
`#include <stdio.h>`	`#include <cstdio>`
	`using namespace std`
`#define PI 3.14`	`const float PI = 3.14`
`#define max(a,b) (...)`	`inline max(a, b) {...}`

Simple I/O functions

```
cout << "Print something" << endl
```

```
cin >> input
```

Cast

static cast : (*safe cast*) Convert if there is a rule based conversion, otherwise error	`static_cast<double> 5/4`
reinterpret cast :	
dynamic cast : used with object	
const cast : cast away const-ness	

Function call

Call by value

Call by pointer

Call by reference

An Introduction to Reference

</>C++ Overload

```
inline void swap(int &i, int &j) {
    int tmp = i
    i = j
    j = tmp
}

inline void swap(double &i, double &j) {
    double tmp = i
    i = j
    j = tmp
}
```

</>C++ Generic

```
template<class T>
inline void swap(T &i, T &j) {
    T tmp = i
    i = j
    j = tmp
}
```

Week 2

Function default parameter

```
T sum (T arr[], int count, T s = 0)
```

</>C++ Multiple template arguments

```
template <class T1, class T2>
void copy (const T1 src[], T2 dest[], int size) {

    for (int i = 0; i < size; ++i) {
        dest[i] = static_cast<T2>(src[i]);
    }

}
```

Enumerate type

Enumeration : auto casted to `int`	`typedef enum {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY} days;`
Enumeration class : need to `static_cast<int>()` explicitly	`enum class WeekDays : int {MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY, SUNDAY};`

«C++ for C Programmer» Notes

by Firmin Martin

</>C++ Operator overloading

```
/* prefix operator */
WeekDay& operator++ (WeekDay &d) {
    return d = static_cast<WeekDay>((static_cast<int>(d) + 1)
 ↪   % 7);
}

/* suffix operator, add extra parameter to distinguish from
 ↪   prefix parameter */
WeekDay operator++ (WeekDay &d, int) {
    WeekDay tmp = d;
    d = static_cast<WeekDay>((static_cast<int>(d) + 1) % 7);
    return tmp;
}
```