# C++ 语法基础 II

尝试用最短时间学会写程序

AtomFirst
南京航空航天大学
2024-10-12

# 目录

## 解决问题的两个步骤

1. Thinking: Problem -> Solution
2. Coding: Solution -> Code

这节课我们解决 `Coding` 的问题

# 怎么学 Coding

- 得到最基本的 `knowledge` （在这节课及上节课）
- 进行有质量的 `practice` （在 OJ 上提交你的代码）
- 现有 `knowledge` 不足后，进行新的 `knowledge-practice 循环` （从书本、网课或者线上教程中学习新知识，然后在 OJ 上进行练习）

# 最小完备程序

## 图灵完备

- 指可解决一切可计算问题
- 算法问题是可计算问题
- 所以图灵完备的语言可解决一切可计算问题
- 我们将在这一节建立一个完备的内部体系，理论上有能力实现一切可能的 `Coding` （但是实际中不够方便，需要改进）

## 数据 + 运算

程序可以分为数据和对数据的运算

# int类型的变量

所有的数据在程序中都可以用整数表示

在 C++ 中用 `int` 表示整数类型

```
int a=5; // 声明了一个int类型的变量，并赋值为5
```

变量是一个盒子，可以往里面放东西

# int 的基本运算

可以对两个 `int` 类型的值进行加减乘除和取模（求余数）

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

int main(){
    int a=3,b=5;
    int c=a+b;
    // int c=a-b;
    // int c=a*b;
    // int c=a/b;
    // int c=a%b;
    return 0;
}
```

## 内部与外部

- 内部：程序核心逻辑
- 外部：调用与世界互动的接口

# 输入输出

虽然我们对 `a` 和 `b` 进行了运算，却不知道结果是什么，因为结果存储在内部

通过 `cout<<` 输出，可以把内部的结果展示到外部

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

int main(){
    int a=3,b=5;
    // cin>>a>>b;
    int c=a+b;
    cout<<c;
    return 0;
}
```

# 分支结构

如果条件不同，则用不同方式处理

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

int main(){
    int x; cin>>x;
    if(x>0){
        cout<<1;
    }else{
        cout<<0;
    }
    return 0;
}
```

# 循环结构

让程序做多次相似的工作

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

int main(){
    int x=0;
    while(x<5){
        cout<<x<<" ";
        x=x+1;
    }
    return 0;
}
```

# 一维数组

可以存储一列数据

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

int main(){
    int a[5];
    for(int i=0;i<5;i++)
        cin>>a[i];
    for(int i=4;i>=0;i--)
        cout<<a[i];
    return 0;
}
```

## 代码复用

不用写多次相同的代码

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

void say_hello(string name){
    cout<<"Hello,"<<name<<"!\n";
}

int main(){
    say_hello("cafebabe");
    say_hello("c10uds");
    return 0;
}
```

# 引用与 swap

本质是一个别名

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

void swap(int &a,int &b){
    int t=a;
    a=b; b=t;
}

int main(){
    int x=3,y=5;
    swap(x,y);
    cout<<x<<" "<<y;
    return 0;
}
```

# 递归

调用自己

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

void print(int x){
    if(x/1000){
        print(x/1000);
        cout<<'_';
    }
    cout<<x%1000;
}

int main(){
    print(1145141919);
    return 0;
}
```

# STL 入门

## `container` & `algorithm`

- `container`：存储多个数据，可以遍历
- `algorithm`：对多个数据进行操作

## `set` 是一种 `container`

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

int main(){
    set<int>st{1,1,4,5,1,4};
    st.insert(19);
    st.erase(19);
    st.insert(810);
    cout<<st.count(1)<<"\n";
    for(auto x:st)
        cout<<x<<" ";
    return 0;
}
```

## `sort` 是一种 `algorithm`

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

int main(){
    vector<int>a{1,1,4,5,1,4,1,9,19,810};
    sort(a.begin(),a.end());
    for(auto &x:a)
        x*=10;
    for(auto x:a)
        cout<<x<<" ";
    return 0;
}
```

# 听完课你应该做的事

- 听课不能直接提高你的编程能力，只能给你关于编程的知识。只有通过练习才能提高你的编程能力
- 每天安排一定的时间（建议一小时以上），进行本门课对应的练习（入门1 到 入门6）
- 遇到问题尝试通过搜索、题解解决，如果自己不能解决可以在招新群中求助同学或学长

# 附：怎么学 ACM

1. 学一点 `C++` （能写就行，参考上面）
2. 多 写 题

# Practice is all you need!