本篇为 CPP 语法教程，主要是为了防止一段时间没刷题后，忘记语法而记录的．不针对深层的语法细节，仅仅是记录机试需要知道的语法知识和相关而已．

入门级代码:

```
#include <cstdio>

int main()
{
    int a, b ;
    scanf("%d%d",&a,&b);
    printf("%d",a+b);
    return 0;
}
```

- C++向下兼容 C 语言，头文件<cstdio>等价于<stdio.h>，类似的如<math.h>和<cmath>也是等价的，具体选择哪一个头文件,看个人喜好. cstdio 是标准输入输出库，程序需要从外界读入和输出数据就必须从该库中调用 scanf 和 printf 函数. C++也有专门的输入输出库 <iostram> 但是其中的输入输出函数 cin, cout，程序执行耗费时间比 printf 大得多，因此机试时最好默认使用 cstdio 库.
- main 函数是程序入口，一个程序仅能有一个主函数.
- scanf 和 printf 是头文件 cstdio 内的函数．两者都需要指定**数据类型**．用%号+缩写字母表示:

数据类型	int	long long	float	double	char	string
格式符	%d	%lld	%f	%lf	%c	%s

scanf 需要通过变量的**指针**将输入值存入变量，用&(取值运算符)表示索引变量指针.

printf 中%d 决定了输入变量的格式为整形(int)，还可以规定输出数据的长度．比如%5d，表示默认输出长度为 5 个字符，不足 5 位则前面补空格，超过 5 位则按%d 处理．还可以使用%05d，表示以 0 补空格，如 123-->00123．浮点数也可以使用相同的方法，如%5.2f，表示总长为 5 个字符串，小数部分占 2 个字符串，由于小数点占一个字符串，因此整数部分仅占 2 个字符串．比如使用%05.2f，则 1.245-->01.25．有的系统是四舍五入(我的电脑就是)，其它电脑可能是"四舍六入五成双"的规则进行小数处理.

数据类型

整型	int	(-2^31~2^31-1)
	long long	(-2^63~2^63-1)
浮点型	float	很大,实际精度 6~7
	double	很大,实际精度 15~16
字符型	char	-128~+127
布尔型	bool	0,1,true,false,
字符串	char []	等价于字符串数组.

可以看出 int 负数范围比正数多 1 个．可以简单认为是 2^31=20 亿=2*10^9=2G.

字符型可直接当 short 整型看待．常见的编程题都会巧妙的控制在 int 内．因此默认选 int 型.

赋值

```
    int a = 1, b = 3;
    char s[25] = "hello world";
```

字符串初始赋值时可直接赋值，后续只能一个一个改.

强制类型转换:

通过括号+类型：如 a = (int) a;

因为整数进行除法时默认为整形除法，因此需要转换为 float 进行乘除.

运算符

普通运算符:

自增++,自减--,求余%.

其中自增++可在元素前后, f(a++)等价于 f(a); a=1+1; f(++a)等价于 a=1+1; f(a);

逻辑运算符:

非: !，与&&，或||,

三目运算:

d= c ? a :b，若 c 为真, d=a，否则 d=b;

位运算:

左移: << ，比如 1 左移 2 位, 1<<2 = 4. 右移>>;

按位于: &, a&b，比如 3&5=3;

按位或: |，比如 4|3=7；实际上就是没有进位的加法.

按位异或: ^，比如 5^3=2;

取反: ~，比如~5=2;

常见判断循环语句

if (判断条件) {正确执行} else{错误执行};

```
    if (a > 3){ cout << "a>3"; }
    else { cout << "a<3";  }
```

switch (样本) { 样本 1: {} 样本 2: {} }

```
    switch (a)
    {
    case 1:    {     cout << "a=1";    }
    case 2:    {     cout << "a=2";    }
    default:    {    cout << " default";}
    }
```

for (初始语句; 退出判断; 底层语句) {执行体;}

```
    for (int i = 3; i < 10; i = i + 2)
    {
        cout << i << endl;
    }
```

while (执行判断) {执行体;}

```
    while (a <0){ a--; }
```

continue 与 break 用在循环体内, break 退出循环, continue 跳出本次循环,进入下次循环.

数组:

类型 + 变量名 [大小]; 注意 string 需要额外一个存储'\0', 因此空间定义比需要的大 1.

```
    int as[5] = {1,2,3,4,5}; //索引 a[0],...,a[4]
    int as[2][3] = {{3},{4,5,5}}; //没被赋值默认为 0;
    char str[15] = {'f','o','o'};
    char ss[15] = 'hello'; //仅在初始化能用
```

大数组需要在 main 函数外[10^6], 不然函数栈会爆掉. 编程题通常小于[10^6];

结构体:

```
struct hello
{
    int val;
    char abc;
    hello(int _val, char _abc){val =_val; abc=_abc};//初始化函数
};
    hello a;
    a.val=3;
    cout << a.val;
```

指针

代表变量指向指针(unsigned int), &代表取址符. 因此&可搭配使用.

```
int a, b;
int *p1 = &a; //p1 为指针变量 等价写法是 int *p; p=&a;
*p1 = 233; //a = 233, 两种写法一致. 另外指针用->代表点来索引.
```

常见的库函数:

记得加上 using namespace std; 不然很麻烦.

数学: math

```
#include <cmath>
   int a = 5, b = 3;
    cout << log(a) << endl;//默认为 e 底,只能通过 log(a)/log(2)得到 2 底的值
    cout << fabs(b - a) << endl;              //去绝对值
    cout << floor(a + 0.3) << ceil(a + 0.5) << round(3.4) << endl;//取整
```

字符串: string

```
    #include <cstring> //有个缺点是默认以空格结束, 可通过正则化解决
    char s1[4] = "app";
    char s2[5] = "baa";
    scanf("%[^\n],s1); //正则表达式
    cout << strlen(s1); //返回 3, 不包括\0;
    strcmp(s1,s2); //字典序排列, 返回<0, =0, >0;
    strcat(s1,s2); //将 s2 接到 s1 后面
    strcpy(s1,s2); //将 s2 复制到 s1, 包括\0;
```

不定长数组 vector

```
#include <vector>
    vector<int> b(5); //默认为 7 个 0. b(7,5)表示 7 个元素, 值都为 5.
    b.push_back(1);
    b.erase(b.begin()+4); //begin 表示初始位置+2 表示第 2 位置,
    b.pop_back(); //默认为尾部数据,
    for (int i=0; i <b.size();i++)
    {        cout << b[i];     }
```

集合 set

```
#include <set>
    set<int> a; //去重且自动排序的好帮手
    a.insert(10);
    a.insert(4);
    set<int>::iterator it = a.find(4);
    a.erase(10); //按值删除, vector 得按迭代器 it 来删除.
    for (it = a.begin();it!=a.end();it++) {
        cout << *it; //除了 vector 和 string 就只能用迭代器来索引了
    }
```

字符串 string. 和 c 语言的 string.h 不同 因为 c++包含更多方程, 一般直接用标准的 string 库.

```
#include <string> //string 并不是 char, 在函数内是局部变量.
    string a = "aa", b = "bb";
    cout << a+b+a[1] << endl; //允许加号操作, 也可以用 printf("%S",s);
    cout << (a>b) << endl; //直接比较也是可以的.
```

```
    a.length();// 等价于 s.size()
    a.substr(0,1); //根据位置返回子串
    cout << a.find("aa")<< endl; //甚至可以找子串，返回第一次出现的地方.
    cout<< a.replace(0,1,"") << endl; //替换，可以等价于删除位置 0,长度 1 的
子串,然后中间插入 s2.
    //如果找不到 find('s')则返回 npos，一个固定的值，需要自行判断
```

map，哈希表

```
#include <map>
    map<char, int> a; //key, value
    a['a'] = 2; //直接添加，不用调用函数
    a.erase(a.find('a')); //因为删除需要 iterator，因此要用 find
    cout << a.size()<<endl;
```

stack 栈

```
#include <stack>
    stack<int> a;
    a.push(1); //加入
    a.pop(); //弹出
    a.top(); //返回头结点
    a.empty(); //返回 flase 或 true，等价 a.size()==0;
```

queue 队列，还有优先队列(大根堆)

```
#include <queue>
    queue<int> a;
    a.push(1);
    cout << a.back() << a.front();
    //返回头结点，末节点，索引还是容易的，只是 pop 只能头部
    a.size(); // 大小，==0 等价于 a.empty()
#include <queue> //map 里面默认 include <utility> 里面有包含 pair
    priority_queue<int> a;
    a.push(3);
    a.push(5);
    a.push(1);
    while (!a.empty())
    {
        cout << a.top() <<endl; //pop 不会返回，只能 top 索引.
        a.pop();
    }
```

pair 类似结构体，但是更简单一些

```
#include <map> //map 里面默认 include <utility> 里面有包含 pair
    pair<int, string> a(5,"string"); //可以这样初始化.
    cout << a.second<<endl; //输出
    cout << (make_pair(5,'f')<make_pair(5,'s')) << endl; //自带比较功能
```

algorithm 算法.

```
#include <algorithm>
    int x=4, y=3;
    int a[5]={1,2};
    abs(x); //abs 的 x 必须整数，cmath 里面 fabs 可以是浮点数
    max(x,y); // min max 都在 algorithm 里面.
    swap(a[0],a[1]); //交换 xy 的值，数组也可以交换. 不错
    cout << a[0]<<a[1]<<endl;

    string b = "hell";
    reverse(b.begin(),b.begin()+2);
    //在 [a,a+2)之间 reverse，其中 a 是迭代器，数组可直接用变量名代替，
    fill(a,a+3,233); // 其中 a 仍为迭代器，表示片段之间填满 233 值
    cout << a[2]<<endl;
```

sort 这也是属于 algorithm 但比较重要，单独讲

```
#include <algorithm>
using namespace std;
bool cmp(int a, int b) //当然可以自定义 int 或者 node vector 类型的比较器.
{
    return a>b; //a 越大越好，因此是从大到小排序
}
    int a[5]={1,2,3,4,5};
    sort(a,a+4,cmp);
 //默认从小到大,cmp 是自定义的比较器,a 是迭代器,通常用 begin, end()得到迭代器
    int length = sizeof(a)/sizeof(int);
    for (int i=0; i<length; i++) //如果不知道长度可用 length;
    {  cout << a[i]<<endl;  }
```

常见的代码

二叉树结构体

```
    struct TreeNode
    {
        int val;
        TreeNode *left;  TreeNode *right;
        TreeNode(int x) : val(x), left(NULL), right(NULL) {}
    };
    TreeNode* left;
    left->val = 3; //指针要用->来索引属性.
```

不定长二维数组

```
int n = 5, m = 3; //变长数组的定义
    vector<vector<int> > a(n); //初始化 5 行
    for (int i = 0; i < n; i++)
    {
        a[i].resize(m); //初始化 3 列
    }
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < m; j++)
        {
            a[i][j] = a[i].size() * i + j; //赋值
            cout << a[i][j] ;
        }
        cout <<endl;
    }
```

参考

- printf 输出格式大全—百度文库 .