小马加编信息学教案(三十七)

STL (map, pair)

一. 课程内容

1. map
2. $pair$

二. 知识讲解

1. map

1. 1 map的介绍

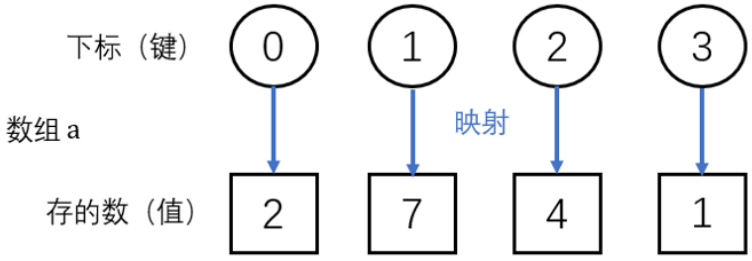
> map翻译为**映射**，是STL中的常用容器。那什么是映射？

> 映射其实就是**键-值对**$(key-value)$的组合。即将**值**存在对应的**键**中，而当我想访问一个**值**的时候只需要访问对应的**键**即可。

> 还是没看懂？其实我们常用的数组也可以理解为一个映射。

> `int a[4];`定义的就是一个映射。比如赋值操作

`a[0] = 2, a[1] = 7, a[2] = 4, a[4] = 1;`



> 其中**键**就是$0,1,2,3$，对应的**值**就是$2,7,4,1$，a数组其实就是将0映射到2，1映射到7，2映射到4，3映射到1。简单理解，数组中的**键**就是对应的下标，**值**就是存在数组中的数。由于数组的下标一定是int类型，所以int数组就能看作是int到int的映射，$char$数组就能看作是int到$char$的映射。

> map就比数组强大多了，它可以是任意类型到任意类型的映射，并且不像数组在定义时就需要确定键值，map可以在**使用过程中任意添加任意键到任意值的映射**，当然为了实现这个功能需要用更多的运行时间作为代价。

> 还是没看懂？那就先把map当作一个下标可以是任意类型的数组吧。

1. 2 map的定义

> 要使用map，必须先添加map头文件，即`#include <map>`，同时必须要有`using namespace std`（使用STL都要加）。

> 定义一个map的方法如下：

```
map<typename1,typename2> name;
```

> 其中，$typename1$是映射前的类型（键，对应数组中的下标），$typename2$是映射后的类型（值，对应数组中存的值），$name$为映射的名字（对应数组名）。

> 普通int数组a就相当于

```
map<int, int> a;
```

> 而对于字符型到字符型映射的map类型lis即为

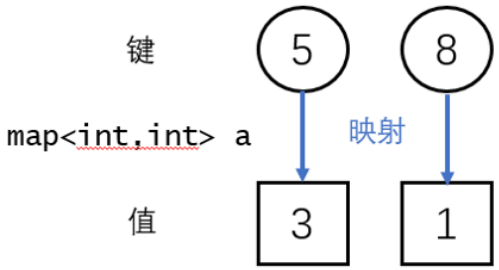
```
map<char, char> lis;
```

1. 3 map的存储

> 对map进行修改时有两种情况。

> 一种是向map中加入映射时，该map中还未存在该键对应的映射，那么就新建立一个映射关系，比如

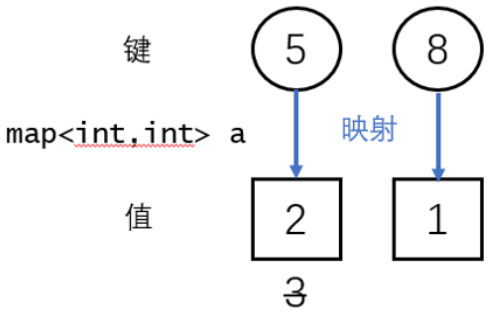
```
map<int, int> a;
a[5] = 3;
a[8] = 1;
```



> 这相当于先在map类型a中加入5到3的映射和8到1的映射。

> 另一种是向map中加入一种映射时，该map已经存在该键的映射，那么直接修改改映射的值即可，比如：

```
map<int, int> a;
a[5] = 3;
a[8] = 1;
a[5] = 2;
```



> 这相当于先在map类型a中加入5到3的映射，然后再将5的映射修改成2，其实就是数组中的修改操作。

1. 4 map的访问

> 若要访问一个键对应的值，直接类似数组，将该键作为map类型的下标即可

```
map<int, int> a;
a[15] = 3;
a[14] = a[15];

map<char, int> lis;
lis['c'] = 3;
lis['c'] = 10;
printf("%c", lis['c'])
```

1. 5 map的时间复杂度

> 假设map存在n对映射关系，那么每次存储和访问的时间复杂度都是$O(log_2^n)$的。

2. $pair$

2. 1 $pair$的介绍

> $pair$翻译为**对**，其功能时将两个元素捆绑起来一起存储，方便代码的编写。

2. 2 $pair$的定义

> 要使用$pair$，必须先添加$pair$的头文件，即`#include <pair>`，同时，必须要有`using namespace std`。

> $pair$有两个参数，分别对应捆绑的两个元素的数据类型，可以是任意基本类型或容器。

```
pair<typename1,typename2> name;
```

> $typename1$是$pair$中第一个元素的类型，$typename2$是$pair$中第二个元素的类型，$name$是该$pair$的名字。

2. 3 $pair$的使用

> 对于$pair$类型变量a，可以通过$a.first$和$a.second$分别对a中第一个元素和第二个元素进行访问。

```
pair<int, int> a;
pair<char, int> b;

a.first = 3;
a.second = 5;
printf("%d", a.first);

b.first = 'c';
b.second = 10;
```

> 对于相同类型的$pair$可以直接进行比较，比较的规则是先比较第一个元素，若相等，再比较第二个元素。

```
pair<char, int> a, b;

a.first = 'd';
a.second = 5;
b.first = 'd';
b.second = 3;
if (a > b) printf("0"); else printf("1");

a.first = 'e';
if (a > b) printf("0"); else printf("1");
```

> 如上述程序，输出的结果将是01。

三. 经典例题

1. 修改数组

> 有一个长度为n的数组a，一开始每个位置的值都为0，现在有m次操作，每次操作给定两个参数x, y，要求输出$a[x]$的值，并将其修改成y。

> 输入格式：
> 第一行两个整数n, m。
> 接下来m行，每行两个整数x, y，意义如题目所述。
> $x \leq n \leq 10^9$
> $m \leq 10^5$
> $0 < y \leq 10^9$

> 输出格式：
> 一共m行，表示m个操作的输出。

样例输入	样例输出
10 5 3 5 4 1 7 1 3 10 7 3	0 0 0 5 1

2. 成绩排名

> 有n个学生，编号分别为$1 \sim n$，编号为i的人的成绩为a_i，将这n个人的编号按成绩从大到小输出。
> 用$pair$实现。

> 输入格式：
> 第一行一个整数n，表示人数。
> 第二行n个整数，第i个整数a_i表示第i个人的成绩。
> $n \leq 1000$
> $a_i \leq 10^9$

> 输出格式：
> 一行n个数，表示这n个人的编号按成绩从大到小输出。

样例输入	样例输出
5 10 7 8 4 6	1 3 2 5 4

四. 提高巩固

1. 修建道路

> 现在有n个城市，一开始城市间没有道路相连，现在有两种操作，第一种是在城市x和城市y（x不等于y）之间修建一条长度为l的道路，另一个是询问城市x, y之间直接相连的最短路径长度是多少。总共有m个操作。

> 输入格式：
> 第一行两个整数n, m
> 接下来m行，每行第一个整数$type$等于1或2表示操作类型。若$type = 1$则为第一种操作，再读入x, y, l表示在城市x, y之间修建一条长度为l的道路。若$type = 2$则为第二种操作，再读入x, y，询问x, y之间直接相连的最短路径长度是多少。
> $n, m \leq 10^5$

> 输出格式：
> 对于第二种操作，输出一个整数表示最短路径长度，若x, y之间暂无道路则输出-1。

样例输入	样例输出
5 6 2 1 2 1 1 2 4 2 1 2 1 3 5 2 1 3 5 1 2 3 5	-1 4 1

2. A - B

> 给出一个有n个数的数组a以及一个数字c，要求计算出所有 $A - B = C$ 的数对 的个数。（注意： 不同位置的数字一样的数对算不同的数对）

> 输入格式：
> 第一行两个整数n， c
> 第二行n个整数，表示数组a中的数
> $n \leq 10^5$
> $a_i \leq 10^9$

> 输出格式：
> 一行一个整数，表示满足题目要求的数对对数。

样例输入	样例输出
6 1 3 4 5 3 3 4	8

3. 找区间

> 给你n个数字，然后给你一个数m，让你求累加和为m的长度最长的连续子串的长度。我们规定数字可正，可负，可零。
> 假如给你一串数字：1，0，0，0，0，7，-1，1，10，5
> m等于7，那么最长的连续子串的长度是7，这个子串是：0，0，0，0，7，-1，1。累加和为7

> 输入格式：
> 第一行两个整数n, m。
> 第二行n个整数，表示给出的n个数字。
> $n \leq 10^5$
> $a_i \leq 10^9$

> 输出格式：
> 一个整数，表示最长的和为m的连续子串长度。

样例输入	样例输出
10 1 0 0 0 0 7 -1 1 10 5	7