C++编程

目录

参考代码


```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    char A[100];
    int i, a[3000], c[3000];
    long long x = 1;
    for (int i = 1; i <= 62; i ++) x *= 2;
    sprintf(A, "%lld", x);
    int Len = strlen(A);
    for (i = Len - 1; i >= 0; i --) a[Len-1-i] = A[i] - '0';
    for (i = 0; i < Len; i ++) c[i] += a[i] * 4;
    for (i = 0; i < Len; i ++)
        c[i+1] += c[i] / 10, c[i] = c[i] % 10;
    if (c[i]) i ++;
    for (i = i - 1; i > 0; i --) cout << c[i];
    cout << c[0] - 1;  //个位数单独处理 2^b-1
    return 0;
 }
```

sprintf 函数

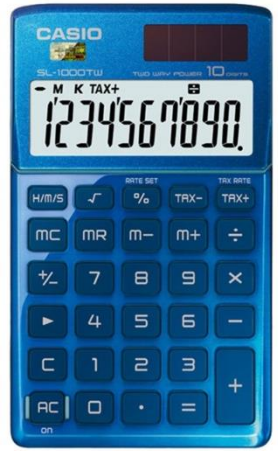
```
9223372036854775807
3.141593
--------------------------------

Process exited after 0.1192 seconds w
ith return value 0
请按任意键继续. . .
```

- sprintf 函数用于把数值转换为字符数组
- 其中占位符的用法和printf一致
- 其实可以把sprintf理解为printf，只不过是输出到字符数组中

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    char s[100], s2[100];
    long long a = 9223372036854775807;
    float b = 3.1415926;
    sprintf(s, "%lld", a);
    sprintf(s2, "%f", b);
    cout << s << endl << s2;
    return 0;
}
```

高精度减法

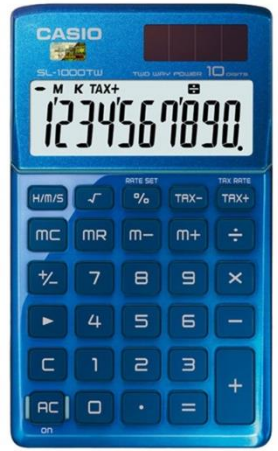


- A-B problem

$0 \le A,B \le 10^{10000}$

Sample input	Sample output
92233720347072921602019020190201902019	720435184450539014 1

高精度减法

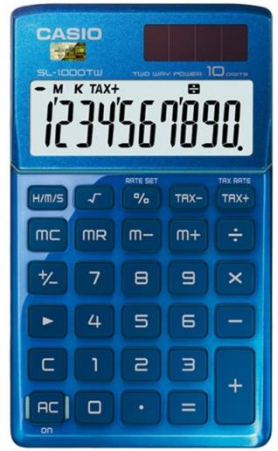


- 高精度减法和加法相类似，只是变进位为借位：

如果不够减，就需要向高位借10，同时高位减1

```
if (a[j] < b[j])
    a[j] += 10, a[j+1] --;
```

高精度减法



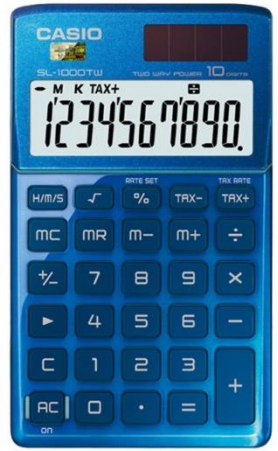
- 同时减法得到的差值有可能为负数，所以还需要特判

但此时的两个运算数都是字符串形态，如何比较大小？

- 我们以前学习字符串的时候，知道字符串是可以直接比较大小的

```
if (A < B)
{
    C = A; A = B; B = C;
    cout << "-";     //先输出负号
}
```

字符串大小的比较



- 比如“123”和“124”这两个字符串，123<124，这与数值大小相符
- 但是“125”和“1221”这两个字符串，125>1221，这与数值大小不符

```
if (A.size() < B.size() || (A.size() == B.size() && (A < B)))
{
    C = A; A = B; B = C;
    cout << "-";    //先输出负号
}
```

参考代码

```
#include<bits/stdc++.h>
using namespace std;
string subtr(string A, string B)
{
    string ans;
    int i, j = 0, a[10010], b[10010], c[10010];
    int LA = A.size(), LB = B.size();
    if (LA < LB || ((LA == LB) && (A < B)))
    {
        ans = A; A = B; B = ans;
        ans = "-";    //处理差可能为负数
     }
    for (i = LA - 1; i >= 0; i --) a[LA-1-i] = A[i] - '0';
    for (i = LB - 1; i >= 0; i --) b[LB-1-i] = B[i] - '0';
    while (j < LA || j < LB)
    {
        if (a[j] < b[j]) {a[j] += 10, a[j+1] --;}   //借位
        c[j] = a[j] - b[j];
        j ++;
    }
    while (!c[j] && j >= 1) j --;   //清理高位零
    for ( ; j >= 0; j -- ) ans += c[j] + '0';
    return ans;
}
```

```
int main()
{
    string A, B;
    cin >> A >> B;
    cout << subtr(A, B);
    return 0;
}
```

结构体

- int 类型可以描述整型数，double类型可以描述浮点数，string可以描述字符串

- 但世间万事万物，有些数据单一类型无法描述，比如学生档案，就涉及姓名(string)、性别(char)、年龄(int)、学籍号(string)、考分(float)、是否获奖(bool)等

结构体

- 那我们希望有数据类型长这样：

姓名	学号
成绩	三好学生

- 然后数组长这样：

a[0]		a[1]		a[2]		a[3]		a[4]		a[5]	
姓名	学号	姓名	学号	姓名	学号	姓名	学号	姓名	学号	姓名	学号
成绩	三好学生	成绩	三好学生	成绩	三好学生	成绩	三好学生	成绩	三好学生	成绩	三好学生

结构体

- 我们知道除了系统中自带的函数以外，可以根据需要自定义函数

- 结构体实际上就是自定义复合数据类型

结构体的定义

- 定义结构体类型：

1. 该数据类型名为student，这个名称与int、float等名称同性质
2. 每一个student类型包含四个域

```
struct student
{
    char name[20];
    char code[10];
    float score;
    bool excellent;
};
```

- 继续定义元素类型为student的数组a：

```
student a[1010];
```

结构体的定义

- 一般写成二合一：

```
struct student
{
    char name[20];
    char code[10];
    float score;
    bool excellent;
} a[1010];
```

结构体类型域的引用

- 在这个例子中，变量a、b都是student类型，要访问其score域，需要使用a.score

```
struct student
{
    char name[20];
    char code[10];
    float score;
    bool excellent;
} a, b;

b.score = a.score;
```

结构体的使用

```
5
张三 20190301 115 1
李四 20190302 97 0
王五 20190303 108 1
赵六 20190304 127 1
陈七 20190305 99 0

张三 20190301
王五 20190303
赵六 20190304
共有以上3位同学入选
--------------------------------
```

```cpp
#include<bits/stdc++.h>
using namespace std;
int n, cnt;
struct student
{
    char name[20];
    char code[10];
    float score;
    bool excellent;
} a[1010];
int main()
{
    scanf("%d", &n);
    for(int i = 0; i < n; i ++)
        scanf("%s%s%f%d", a[i].name, a[i].code, &a[i].score, &a[i].excellent);
    for (int i =0; i < n; i ++)
        if (a[i].excellent)
        {
            printf("%s %s\n", a[i].name, a[i].code);
            cnt ++;
        }
    printf("共有以上%d位同学入选", cnt);
    return 0;
}
```

结构体的使用

```
5
张三 20190301 115 1
李四 20190302 97 0
王五 20190303 108 1
赵六 20190304 127 1
陈七 20190305 99 0

张三 20190301
王五 20190303
赵六 20190304
共有以上3位同学入选
--------------------------------
```

- 从这个例子我们可以看出：

结构体的最大意义在于把相关联的不同单一数据类型组合在一起

谁拿了最多奖学金

- 某校的惯例是在每学期的期末考试之后发放奖学金。发放的奖学金共有五种，获取的条件各自不同：
- 院士奖学金，每人8000；五四奖学金，每人4000；成绩优秀奖，每人2000；西部奖学金，每人1000；班级贡献奖，每人850，详见luogu1051
- 只要符合条件就可以得奖。现在给出若干学生的相关数据，请计算哪些同学获得的奖金总数最高

Sample input	Sample output
4 YaoLin 87 82 Y N 0 ChenRuiyi 88 78 N Y 1 LiXin 92 88 N N 0 ZhangQin 83 87 Y N 1	ChenRuiyi 9000 28700

分析

- 把相关信息存入结构体数组后，在结构体数组中维护最值即可

```
struct node
{
    char name[21];    //姓名
    int pj;    //期末平均成绩
    int py;    //班级评议成绩
    char gb;   //学生干部
    char xb;   //西部省份
    int lw;    //论文
    int money = 0;  //该生所得奖金
} a[110];
```

参考代码

```
#include<bits/stdc++.h>
using namespace std;
int n, sum, maxm, pos;    //sum、maxm、pos记录奖金总额、最高奖金额及序号
struct node
{
    char name[21];    //姓名
    int pj;    //期末平均成绩
    int py;    //班级评议成绩
    char gb;   //学生干部
    char xb;   //西部省份
    int lw;    //论文
    int money = 0;  //该生所得奖金
} a[110];
```

```
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; i ++)
    {
        scanf("%s%d%d%c%c%d", a[i].name, &a[i].pj, &a[i].py, &a[i].gb, &a[i].xb, &a[i].lw);
        if (a[i].pj > 80 && a[i].lw >= 1) a[i].money += 8000;  //院士奖学金
        if (a[i].pj > 85 && a[i].py > 80) a[i].money += 4000;  //五四奖学金
        if (a[i].pj > 90) a[i].money += 2000;   //成绩优秀奖
        if (a[i].pj > 85 && a[i].xb == 'Y') a[i].money += 1000;   //西部奖学金
        if (a[i].py > 80 && a[i].gb == 'Y') a[i].money += 850;    //班级贡献奖
        sum += a[i].money;
        if (a[i].money > maxm)
            pos = i, maxm = a[i].money;
    }
    printf("%s\n%d\n%d", a[pos].name, maxm, sum);
    return 0;
}
```

Sort 函数

```
5
2 1 3 5 4
1 2 3 4 5
--------------------------------
Process exited after 4.179 seconds wi
th return value 0
请按任意键继续. . .
```

- Sort 函数用于给数组元素排序，默认从小到大

```cpp
#include<bits/stdc++.h>
using namespace std;
int a[10010], n;
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; i ++) scanf("%d", &a[i]);
    sort(a, a + n);
    for (int i = 0; i < n; i ++) printf("%d ", a[i]);
    return 0;
 }
```

Sort 函数

```
5
2 1 3 5 4
5 4 3 2 1
--------------------------------

Process exited after 5.232 seconds wi
th return value 0
请按任意键继续. . .
```

- 如果要从大到小？

```cpp
#include<bits/stdc++.h>
using namespace std;
int a[10010], n;
bool cmp(int a, int b)
{
    return a > b;
}
int main()
{
    scanf("%d", &n);
    for (int i = 0; i < n; i ++) scanf("%d", &a[i]);
    sort(a, a + n, cmp);
    for (int i = 0; i < n; i ++) printf("%d ", a[i]);
    return 0;
 }
```

Sort 函数

- 这个写法：

```
return a > b;
```

- 等价于：

```
if (a > b) return true;
else return false;
```

Sort 函数

- 现实需求中常有更复杂的应用：比如足球比赛中，排名先看积分，积分相同时，则看净胜球/进球数/胜负关系等等；又比如成绩排名，以成绩为第一排序关键词，成绩相同的同学，学号小的靠前

- 我们现在以携带物品为例：优先携带价格高的，如果价格一样，则携带重量轻的

Sort 函数

- 显然我们需要开结构体数据类型来表示一个物品

```
struct pack
{
    int value;  //价格
    int weight;  //重量
} a[10010];
```

Sort 函数

```
5
180 5 160 4 100 2 160 5 180 6

180 5
180 6
160 4
160 5
100 2
```

```cpp
#include<bits/stdc++.h>
using namespace std;
struct pack
{
    int value;
    int weight;
} a[10010];
bool cmp(pack x, pack y)
{
    if (x.value == y.value) return x.weight < y.weight;    //等价值则重量轻在前
    return x.value > y.value;
}
int main()
{
    int n; scanf("%d", &n);
    for (int i = 0; i < n; i ++)
        scanf("%d%d", &a[i].value, &a[i].weight);
    sort(a, a + n, cmp);
    for (int i = 0; i < n; i ++)
        printf("%d %d\n", a[i].value, a[i].weight);
    return 0;
 }
```

Sort 函数

- 这个写法：

```
bool cmp(pack x, pack y)
{
    if (x.value == y.value) return x.weight < y.weight;
    return x.value > y.value;
}
```

- 等价于：

```
bool cmp(pack x, pack y)
{
    if (x.value > y.value) return true;
    else if (x.value == y.value)
    {
        if (x.weight < y.weight) return true;
        else return false;
    }
    return false;
}
```

分数线划定

- 为了选拔最合适的人才，A市对所有报名的选手进行了笔试，笔试分数达到面试分数线的选手方可进入面试。面试分数线根据计划录取人数的150%划定，即如果计划录取m名志愿者，则面试分数线为排名第m×150%（向下取整）名选手的分数，而最终进入面试的选手为笔试成绩不低于面试分数线的所有选手
- 现在就请你编写程序划定面试分数线，并输出所有进入面试的选手的报名号和笔试成绩（参加人数至多5000人）

Sample input	Sample output
6 3 //参加人数和录取人数 1000 90 3239 88 2390 95 7231 84 1005 95 1001 88	88 5 //分数线和面试人数 1005 95 2390 95 1000 90 1001 88 3239 88

分析

- 开结构体数组存储每个考生的信息
- 注意处理好分数并列的情况就可以

参考代码

```
#include<bits/stdc++.h>
using namespace std;
struct node
{
    int number, score;
} a[5010];

int cmp (node x, node y)
{
    if (x.score == y.score) return x.number < y.number;
    return x.score > y.score;
}
```

```
int main()
{
    int n, m; scanf("%d%d", &n, &m);
    for (int i = 0; i < n; i ++)
        scanf("%d%d", &a[i].number, &a[i].score);
    m = floor(m * 1.5);  //向下取整
    sort(a, a + n, cmp);
    int line = a[m-1].score;   //分数线
    for (int i = m; i < n; i ++)
        if (a[i].score == line) m ++;
            else break;   //处理分数并列的情况
    printf("%d %d\n", line, m);  //分数线和人数
    for (int i = 0; i < m; i ++)
        printf("%d %d\n", a[i].number, a[i].score);
    return 0;
}
```

期末成绩单

- 期末考试结束了，某门课程的科任老师要制作一份期末排名表。要求是这样的：成绩单以学号为序输出每个同学的成绩排名。但为了隐私，要求隐去其他信息（学生至多2000人，不考虑成绩并列）

Sample input	Sample output
5 //学生人数 98 //考试分数，下同 132 116 79 124	4 1 3 5 2

index	score	rank
1	98	4
2	132	1
3	116	3
4	79	5
5	124	2

分析

- 如果用多个一维数组，排序时没法保持一一对应的关系
- 开结构体数组用于存储学生的信息

```
struct node
{
    int index, score, rank;
} a[2010];
```

index	score	rank

分析

- 读入时，不仅可以获得score，还要保存输入顺序index
- 注意为了和学号对应，我们从1开始

```
for (int i = 1; i <= n; i ++)
{
    scanf("%d", &a[i].score);
    a[i].index = i;
}
```

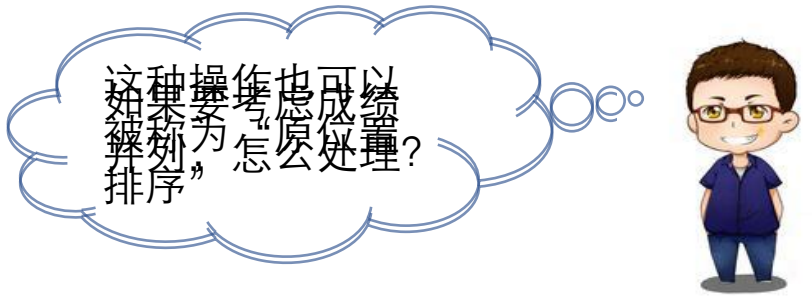
index	score	rank
1	98	
2	132	
3	116	
4	79	
5	124	

分析

- 然后按score排序一次，获得rank
- 再按index排序一次，得到（最终的成绩单）输出顺序

index	score	rank
1	98	4
2	132	1
3	116	3
4	79	5
5	124	2

参考代码




```cpp
#include<bits/stdc++.h>
using namespace std;
struct node
{
    int index, score, rank;
} a[2010];
bool cmpscore(node x, node y)   //按成绩排序
{
    return x.score > y.score;
}
bool cmpindex(node x, node y)   //按学号排序
{
    return x.index < y.index;
}
```

```cpp
int main()
{
    int n; scanf("%d", &n);
    for (int i = 1; i <= n; i ++)
    {
        scanf("%d", &a[i].score);
        a[i].index = i;
    }
    sort(a + 1, a + n + 1, cmpscore);
    for (int i = 1; i <= n; i ++)
        a[i].rank = i;  //此时的a[]已按score排好序
    sort(a + 1, a + n + 1, cmpindex);
    for (int i = 1; i <= n; i ++)
        printf("%d\n", a[i].rank);
    return 0;
}
```

课外加练

- luogu 2142　　高精度减法
- luogu 1051　　谁拿了最多奖学金
- luogu 1068　　分数线划定