ACM 算法模版

辛济远

2017 年 4 月 9 日

目录

1 STL 库

1.1 数据结构

向量 vector

可以改变存储长度的数组。一般用于邻接表等，但由于存储较为耗时，使用时应当注意。

- 声明：`#include`<vector>
- 定义：`vector<`数据类型`>` 数组名;
- 操作：可以按照访问数组的方式直接访问数组中的数据。
  `Q.push_back(`数据`)` 将数据加入数组。
  `Q.pop_back()` 删除数组中最后一个数据。
  `Q.erase(it)` 删除指定地址的数据。
  `Q.clear()` 清除数组中的所有数据。
  `Q.size()` 返回数组的大小。
  `Q.begin()` 返回数组的首地址。
  `Q.end()` 返回数组的尾地址 + 1。
- 迭代器（遍历数组中的元素）:

```
vector<node> Q;
vector<node>::iterator it;
for(it = Q.begin(); it != Q.end(); it++)
  cout << *it << end;
```

队列 queue

一般用于搜索，输出等。无法使用迭代器。

- 声明：`#include<queue>`
- 定义：`queue<`数据类型`>` 队列名;
- 操作：
  `Q.push(`数据`)` 将数据入队；
  `Q.front()` 返回一个定义的数据类型的数据，为队头元素；
  `Q.empty()` 返回 True 或 False，True 为队列为空；
  `Q.pop()` 弹出队头元素，无返回值；`Q.size()` 返回一个整数，为队列中元素个数。

栈 stack

一般用于递归回溯等。无法使用迭代器。

- 声明：`#include<stack>`
- 定义：`queue<`数据类型`>` 队列名;
- 操作：
  `Q.push(`数据`)` 将数据入栈；
  `Q.top()` 返回一个定义的数据类型的数据，为栈顶元素；
  `Q.empty()` 返回 True 或 False，True 为栈为空；
  `Q.pop()` 弹出栈顶元素，无返回值；`Q.size()` 返回一个整数，为栈中元素个数。

链表 link

集合 set

集合 map

1.2 算法

2 高精度模版

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

class BigNum {
public:
  int s[1000];
  BigNum () {
    memset(s , 0 , sizeof(s));
  }
  BigNum operator + (BigNum x) {
    BigNum ans;
    int t;
    ans.s[0] = max(x.s[0] , s[0]);
    t = 0;
    for (int i = 1; i < ans.s[0]; i++) {
      ans.s[i] = x.s[i] + s[i] + t;
      t = ans.s[i] / 10000;
      ans.s[i] %= 10000;
    }
    if (t)
      ans.s[ans.s[0]++] = t;
    return ans;
  }
  BigNum operator * (BigNum x) {
    BigNum ans;
    ans.s[0] = s[0] + x.s[0];
    for (int i = 1; i < s[0]; i++)
      for (int j = 1; j < x.s[0]; j++)
        ans.s[i + j - 1] += s[i] * x.s[i];
    int t = 0;
```

```
    for (int i = 1; i <= ans.s[0]; i++) {
      ans.s[i] += t;
      t = ans.s[i] / 10000;
      ans.s[i] %= 10000;
    }
    while (t) {
      ans.s[ans.s[0]++] = t % 10000;
      t /= 10000;
    }
    return ans;
  }
};

ostream& operator <<(ostream &out, const BigNum &x) {
  out << x.s[x.s[0] - 1];
  for (int i = x.s[0] - 2; i >= 1; i--)
    out << setw(4) << setfill('0') << x.s[i];
  out << endl;
  return out;
}

istream& operator >>(istream &in, BigNum &x) {
  char s[4000];
  in >> s;
  int l = strlen(s);
  x.s[0] = 1;
  for (int i = l - 1; i >= 0; i -= 4) {
    x.s[x.s[0]] = 0;
    for (int j = 0; j < 4 && (i - j) >= 0; j++) {
      x.s[x.s[0]] *= 10;
      x.s[x.s[0]] += s[i - j] - '0';
    }
    x.s[0]++;
  }
 return in;
}
```

```
int main(int argc, const char * argv[]) {
  BigNum a, b;
  cin >> a >> b;
  cout << a + b << endl;
  return 0;
}
```