数据结构 A
作业 2 参考答案

作业情况

教材：数据结构教程（C++ 语言描述）李春葆等
题目范围：顺序表与单链表
邮箱：wjyyy1@126.com
授课教师：彭蓉 教授
助教：王骏峣

习题 1 单选题

下列对顺序存储的有序表 (长度为 n) 实现给定操作的算法中平均时间复杂度为 $O(1)$ 的是（　　）。
A. 查找包含指定值元素的值
B. 插入包含指定值元素的算法
C. 删除第 i 个元素的算法
D. 获取第 i 个值的算法

答案：D
解析：顺序表（顺序存储）：方便随机访问，不方便插入和删除。
链表（链式存储）：方便插入和删除，不方便随机访问。
顺序存储的有序表即有序数组。
A 中，在有序数组中查找包含指定值的元素的位置最快的方法是用二分查找，时间复杂度为 $O(\log n)$。
B 中，在有序数组中查找包含指定值的元素最快的方法是首先用二分查找找到插入元素的位置，时间复杂度为 $O(\log n)$，然后把位于插入元素后面的元素的位置后移一格，平均时间复杂度为 $O(n)$。
C 中，删除第 i 个元素的最快的方法是首先利用顺序表支持随机访问的性质

找到第 i 个元素并删除，时间复杂度为 $O(1)$，然后把位于插入元素后面的元素的位置前移一格，平均时间复杂度为 $O(n)$。

D 中，获取第 i 个元素利用顺序表支持随机访问的性质找到第 i 个元素并访问即可。

习题 2 单选题

以下关于单链表叙述正确的是（　　）

i. 结点除自身信息以外还包括指针域，存储密度小于顺序表

ii. 找第 i 个结点的时间为 $O(1)$

iii. 在插入、删除运算时不必移动结点

A. 仅 i、iii

B. 仅 i、ii

C. 仅 ii、iii

D. 仅 i、ii、iii

答案：A

解析：i 项正确，顺序表存储时可以直接用下标记录相对位置，而单链表需要存储“下一个位置”的信息。

ii 项错误，找第 i 个结点的平均时间为 $O(n)$，因为单链表只能从表头开始查找 i 次，不支持随机访问。

iii 项正确，插入、删除运算时不必移动结点，只需要修改指针即可，原来的数据可以保持不变。

习题 3 设计链表

【问题描述】

请设计一个整数链表包含如下操作（含 n 个结点的链表中结点的序号或者索引为 $0 \sim n-1$）：

`1 val`：在链表的第一个元素之前添加一个值为 val 的结点，插入后新结点将成为链表的第一个结点。

`2 val`：将值为 val 的结点添加到链表的最后一个结点之后。

`3 i val`：在链表中的第 i 个结点之前添加值为 val 的结点。如果 i 等于链表的长度，则该结点将附加到链表的末尾。如果 i 大于链表长度，则不会插入结点。如果 i 小于 0，则在头部插入结点。

`4 i`：如果索引 i 有效，则删除链表中的第 i 个结点。

`5 i`：获取链表中第 i 个结点的值并且输出，如果索引 i 无效则返回 -1。

【输入形式】

每个测试用例由一系列的操作过程组成，以 -1 表示结束。

【输出形式】

对于每个测试用例，若测试用例中有操作 5，则在一行中输出对应的整数。

【样例输入】

```
1 1
2 2
2 3
2 4
2 5
2 7
3 5 6
5 0
5 1
5 2
5 3
5 4
5 5
5 6
4 0
4 1
5 1
5 10
-1
```

【样例输出】

```
1
2
3
4
5
6
7
4
-1
```

【样例说明】

测试用例中的每一行代表一个功能，第一个数字为功能编号，后面的数据为该功能的参数。比如 `3 5 6`，表示的是第 3 个功能，在链表中的第 5 个结点之前添加值为 6 的结点。测试数据的文件名为 `in.txt`。

【评分标准】

该题目有 5 个测试用例，每通过一个测试用例，得 20 分。

答案:

```cpp
#include<iostream>
using namespace std;
struct LinkNode              //单链表结点类型
{
    int data;            //存放int类型数据元素
    LinkNode* next;              //下一个结点的指针
    LinkNode():next(NULL) {}     //构造函数
    LinkNode(int d):data(d),next(NULL) {}
};
int main()
{
    LinkNode *head=new LinkNode();
    int op,x;//输入操作符，如果为-1就停止
    ifstream fin("in.txt"); //使用文件流从in.txt读取数据
    fin>>op;
    while(op!=-1)
    {
        if(op==1)//op=1表示头插法
        {
            fin>>x;
            LinkNode* s=new LinkNode(x);
            s->next=head->next;
            head->next=s;
        }
        if(op==2)//op=2表示尾插法
        {
            fin>>x;
            LinkNode* p=head;
            while(p->next!=NULL)//直到找到尾节点
                p=p->next;
            p->next=new LinkNode(x);
        }
        if(op==3)//op=3表示在第i节点前插入
        {
            int i;
            fin>>i>>x;
            LinkNode* p=head;//找到第i-1节点
            for(int j=0;j<i&&(p!=NULL);j++)
                p=p->next;//超过尾节点也会停下
            if(p!=NULL)//如果超过尾节点，就不插入
```

```
            {
                LinkNode* s=new LinkNode(x);
                s->next=p->next;
                p->next=s;
            }
        }
        if(op==4)//删除第i节点
        {
            int i;
            fin>>i;
            LinkNode* p=head;//找到第i-1节点
            for(int j=0;j<i&&(p->next!=NULL);j++)
                p=p->next;//到达尾节点也会停下
            if(p->next!=NULL)//如果到达尾节点，就不操作
                p->next=p->next->next;
        }
        if(op==5)//输出第i节点
        {
            int i;
            fin>>i;
            LinkNode* p=head;
            for(int j=0;j<=i&&(p!=NULL);j++)
                p=p->next;//到达尾节点也会停下
            if(p!=NULL)//如果到达尾节点，就不操作
                cout<<p->data<<endl;
            else
                cout<<-1<<endl;
        }
        fin>>op;
    }
    return 0;
}
```

解析：参考课件第二讲中的 2.3.2 单链表，建立一个带头结点的单链表，然后根据输入的操作符进行操作。

操作 1 为头插法，操作 2 为尾插法，操作 3 为插入，操作 4 为删除，操作 5 为输出。注意尾节点不要越界，即 p 不为空时才能操作当前节点、p->next 不为空时才能操作下一节点。

考虑到本题中数据均为 int，可以把课件中的 <T> 去掉，直接用 int data 代替。或者按课件中写法，将 LinkNode<T> 换成 LinkNode<int>。

习题 4 两数之和

【问题描述】

给定一个整数数组 $nums$ 和一个目标值 $target$，请你在该数组中找出和为目标值的那两个整数，并返回他们的数组下标。例如，给定 $nums = [2, 7, 11, 15]$，$target = 9$，返回结果为 $[0, 1]$。若没有答案，则返回 $[-1, -1]$。题目要求设计 `twoSum()` 函数：

```
class Solution {
    public:
        vector<int> twoSum(vector<int>& nums, int
            target)
        { ... }
};
```

【输入形式】

每个测试用例由两行数据构成，第一行给出整数数组，元素之间以空格隔开，可以有重复元素；第二行给出求和的目标值。

【输出形式】

返回累加和为目标值的两个整数在数组中的下标值，不限顺序。若没有答案，则返回 $[-1, -1]$。

【样例输入】

```
2 7 11 15
9
```

【样例输出】

```
0 1
```

【样例说明】

输入的整数数组有四个整数，分别为 $2, 7, 11, 15$，目标值为 9。数组中的 2 和 7 的累加和为 9，因此返回这两个整数在数组中的下标地址，即 $0, 1$。测试数据文件名为 `in.txt`。

【评分标准】

共 10 个测试用例，每通过一个测试得 10 分。

答案：

```cpp
#include<iostream>
#include<vector>
#include<fstream>
#include<sstream>
using namespace std;
class Solution {
public:
    vector<int> twoSum(vector<int>& nums, int target) {
        vector<int> ans;
        for(int i=0;i<nums.size();i++)
            for(int j=i+1;j<nums.size();j++)
                if(nums[i]+nums[j]==target)
                {
                    ans.push_back(i);
                    ans.push_back(j);
                    return ans;
                }
        ans.push_back(-1);
        ans.push_back(-1);
        return ans;
    }
};
int main()
{
    int n,x,target;
    vector<int> num;
    ifstream fin("in.txt");
    //使用文件流从in.txt读取数据
    string line;
    getline(fin,line);
    //因为要读入一行，所以使用getline函数，再用fin读入line这一字符串
    stringstream ss(line);
    //再将这一字符串转化为字符串流，用类似的>>方法读入数据
    while(ss>>x)
        num.push_back(x);
    //target只有一个数，所以直接用fin读入即可
    fin>>target;
    Solution s;
    vector<int> ans=s.twoSum(num,target);
    cout<<ans[0]<<" "<<ans[1];
    return 0;
}
```

解析：使用二重循环，找到两个相加等于 target 的元素即可，注意不要找到相同位置的数。

本题可以不使用 `vector` 返回结果，把解题过程写在主函数中更加简便。

对于输入流的常见用法，可以引入 `<fstream>`，使用 `ifstream fin("in.txt")` 定义 `in.txt` 对应的流。

使用 `fin>>x` 可以向 `x`（可为数值或字符串）读入一段数据，直到下一个空格/换行为止，因此可以读入以空格分开的数字或字符串。

使用 `getline(fin,line)` 可以向 `line`（应为字符串）读入一段数据，直到换行为止，这时一整行数据都在 `line` 中。

此时可以直接将 `line` 作为字符串利用，如果要再读入 `line` 中的数据，可以引入 `<sstream>`，使用 `stringstream ss(line)` 定义 `line` 对应的流。

总结

题目：线性表 1

日期：2024 年 3 月 29 日

批改人：王骏峣

邮箱：wjyyy1@126.com

习题 1、2：按书上和课件上内容理解单链表和顺序表的区别和使用。

习题 3：本题以单链表实现为主，可参考课件或书本，出现问题及时跟着报错信息进行尝试。部分同学出现了代码 90% 以上一致的情况，这些同学写完后一定要把这些内容落实为自己的知识。后续作业再出现完全一致的情况将扣除相应题目的分数。

习题 4：注意本题不要找到相同位置的数，总体较为容易。后续增加了文件输入输出的指导，在之后的作业题目中可以参考。感谢张智皓同学提出的建议。

各位同学如有问题欢迎及时在群里提出，或者通过邮件/QQ 联系我。