1. 单项选择题

ABCBD ACBDC

1. 简答题
2. 链式存储结构。原因：线性表在处理过程中长度会动态地变化，说明对线性表的操作使以增加和删除数据为主。而顺序存储结构在增加和删除数据的时候需要整体移动数据的位置，比较复杂，所以宜采用链式存储结构。
3. 最少有73个，最多有235个。

（1）矩阵中不为0的元素的二分之一。

（2）矩阵中对应该顶点的行或列中非零元素的个数。

（3）矩阵中i对应的行和j对应的列的交点元素是否为0。为0的话就不存在边，不为0则存在边。

4. ①折半插入排序比较次数取决于每一趟的折半次数，而折半次数只取决于元素个数而与序列的初始状态无关。②当排序序列元素个数较小时。

三、综合题

1. 交换双向链表中某个指定结点与其直接后继结点的位置。

2. 三个。如下图

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  | 6 |  |  |  |
|  | 2 |  | 3 |  | 4 |  | 5 | 5 |  |  |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |  |

3 .前序：ABDEGCFH 后序：DGEBHFCA

1. 深度优先：ACBDE 广度优先：ACEBD

四 、算法设计题

void insertItem(keytype s[], keytype item, int n)

{

int low, high, mid, j;

low = 0;

high = n - 1;

while (low <= high) {

mid = (low + high) / 2;

if (item < s[mid]) {

high = mid - 1;

} else {

low = mid + 1;

}

}

j = n - 1;

while (j >= low) {

s[j + 1] = s[j];

j--;

}

s[low] = item;

}

五 、单项选择题

ABCBD ACADD

六 、综合题

int i = 0, j = 100, count = 0;

while (i < 100) {

while (j >= i) {

count += j - i;

j -= 2;

}

i++;

}

1. ①msg1是指针变量，表示的是地址，声明的时候不需要指明大小，需要加\*表示这是指针变量。msg2是数组名，是标识符，表示的是整个数组（虽然值也是地址），声明的时候需要指明数组的大小。②使用时要注意：在声明完成之后，msg1可以直接用字符串常量赋值，而msg2只能使用字符常量单独为每个元素赋值。
2. ①fscanf是格式化读入函数，可以指定读入数据的格式，读入的是ASCII字符，在读入的时候需要进行二进制与ASCII字符的转换。Fread是二进制方式读入函数，读入的是二进制数据，不需要做转换。②在不需要频繁与磁盘交换数据的情况下宜采用fscanf函数，使用方便，易于理解。反之宜采用fread函数，效率高。
3. 12480

七 、程序设计题

int freqs(char source[], char destination[])

{

int ns = strlen(source), nd = strlen(destination), i = 0, num = 0;

char str[100];

while (i + nd <= ns) {

strncpy(str, source + i, nd);

if (strcmp(destination, str) == 0) num++;

i++;

}

return num;

}

八 、程序设计题（水平有限，仅实现功能，写的不好请见谅）

#include<stdio.h>

/\*判断是是否是合法的核苷酸序列字符\*/

int atgc(char c)

{

int i = 0;

char str[] = "ATGCatgc";

while (str[i] != '\0') {

if (c == str[i]) return 1;

i++;

}

return 0;

}

/\*去除换行符\*/

void del(char str[])

{

int n = strlen(str);

if (str[n - 1] == '\n') str[n - 1] = '\0';

}

int first(char first[])

{

return first[0] == '@';

}

int second(char second[])

{

int i = 0, n;

if (strlen(second) > 2000) return 0;

while (second[i] != '\0') {

if (!atgc(second[i])) return 0;

i++;

}

return 1;

}

int third(char third[])

{

return third[0] == '+';

}

int fourth(char fourth[], char second[])

{

if (strlen(fourth) != strlen(second)) return 0;

int i = 0;

while (fourth[i] != '\0') {

if (fourth[i] < 33 || fourth[i] > 126) return 0;

i++;

}

return 1;

}

int main(int argc, char \*argv[])

{

if (argc != 2) {

printf("error");

exit(0);

}

FILE \*fp;

int i, j = 1, t;

char s[4][2002]; //序列最长2000加'\0'和'\n'是2002

fp = fopen(argv[1], "r");

if (fp == NULL) {

printf("cannot open this file\n");

exit(0);

}

while (1) {

i = 0;

while (i < 4) {

t = fgets(s[i], 2001, fp);

del(s[i]);

if (t == 0 && j == 1 && i == 0) {

fclose(fp);

printf("文件为空");

return 0;

} else if (t == 0 && i != 0) {

fclose(fp);

printf("不合法");//剩余行数不足一个单元

return 0;

} else if (t == 0 && j != 1 && i == 0) {

fclose(fp);

printf("合法");

return 0;

}

i++;

}

if (!first(s[0])) {

fclose(fp);

printf("不合法");

return 0;

}

if (!second(s[1])) {

fclose(fp);

printf("不合法");

return 0;

}

if (!third(s[2])) {

fclose(fp);

printf("不合法");

return 0;

}

if (!fourth(s[3], s[1])) {

fclose(fp);

printf("不合法");

return 0;

}

j++;

}

}