switch (表达式) {

case 常量1: // 如果表达式的值是常量1，执行此代码

break;

case 常量2: // 如果表达式的值是常量2，执行此代码

break;

case 常量3: // 如果表达式的值是常量3，执行此代码

break;

default: // 如果没有匹配的常量，执行此代码

}

remove: 这是一个算法，它会在给定范围内移除所有与给定值相等的元素。remove 实际上并不会从容器中删除元素，而是将所有非目标元素移到容器的前面，返回一个指向新 "逻辑" 末尾的迭代器。它的签名通常是：

std::remove(iterator first, iterator last, const T& value);

它将范围 [first, last) 中的所有 value 删除，并返回一个迭代器，指向删除后新逻辑末尾的位置。

erase: 这是 std::vector 提供的成员函数，实际上会从容器中删除元素。erase 接受两个参数：一个指向起始位置的迭代器和一个指向结束位置的迭代器。

组合使用时，remove 会把指定值移到容器的末尾并返回新末尾的位置，而 erase 则会把这些位置的元素实际从容器中移除。

M.erase(remove(M.begin(), M.end(), M[0]), M.end()):  
这行代码做了以下两件事：

remove(M.begin(), M.end(), M[0]) 会找出 M 中所有与 M[0] 相等的元素，并将这些元素移到容器的末尾。返回的是一个指向新逻辑末尾的迭代器。

1. erase(...) 会真正地删除这些元素。

整体上，这段代码的作用是：

1. 在 M 容器不为空时循环。
2. 每次循环，从 M 中取出第一个元素 M[0]。
3. 在 N 中删除所有与 M[0] 相同的元素。
4. 在 M 中删除所有与 M[0] 相同的元素。

最终，这段代码会删除 M 和 N 中所有相同的元素，直到 M 为空。

### stringstream ss(time);

stringstream 是 C++ 标准库中的一个类，位于 <sstream> 头文件中。它是一个字符串流类，可以用来将数据从字符串中提取出来，或者将数据写入字符串中。它有类似于输入流和输出流的功能。

time 是传入的字符串变量，假设它代表某个时间值（例如 "12:34"）。

ss 是 stringstream 类型的对象，它通过构造函数初始化，将 time 字符串加载到流中。

### 2. ss >> hours >> colon >> minutes;

这行代码使用了流操作符（>>）从字符串流中提取数据。流操作符 >> 允许从流中提取并解析数据，直到遇到空格、换行或其他非期望的字符为止。

ss >> hours:  
这是将流中的数据提取并存储到 hours 变量中。假设 time 字符串的前两位代表小时（例如 "12:34" 中的 12），这些字符将被提取并转换为整数存储在 hours 变量中。

ss >> colon:  
colon 变量用于接收分隔符（即 ":"）。这意味着流会在 hours 后期望一个冒号字符，所以它会将这个冒号字符提取并存储在 colon 变量中。如果 time 字符串中没有冒号或它不符合预期格式，流提取将失败。

ss >> minutes:  
类似地，剩余的字符（假设为分钟部分，如 "12:34" 中的 34）将被提取并转换为整数，存储到 minutes 变量中。

因此，这一行的作用是从字符串流中依次提取小时、冒号和分钟值，并分别存储到 hours、colon 和 minutes 变量中。注意，colon 的内容不被用作后续计算，只是为了从流中跳过冒号字符。

### 3return hours \* 60 + minutes;

这行代码是计算时间的总分钟数。假设 hours 是小时数，minutes 是分钟数，公式 hours \* 60 + minutes 就是将小时转换为分钟后再加上剩余的分钟数。

**#define  \_CRT\_SECURE\_NO\_WARNINGS**

**ESC O ESC O ESC B**

**示例代码：**

排序算法实例

冒泡排序：

#include<stdio.h>

int main()

{

int n; //整数个数

int buf[100]; //整数数组

scanf("%d", &n); //输入n

for (int i = 0; i < n; i++){ //输入n个整数

scanf("%d", &buf[i]);

}

//排序过程

for (int i = 0; i < n; i++) { //外循环为排序趟数，len个数进行len-1趟

for (int j = 0; j < n - 1 - i; j++) { //内循环为每趟比较的次数，遍历比较n-1-i次

注意：j<n-1-i中的i受初始值的影响，当i初值为start，此处i改为i-start

if (buf[j] > buf[j + 1]) { //位置交换条件

int temp = buf[j];

buf[j] = buf[j + 1];

buf[j + 1] = temp;

}

}

}//从小到大

//输出

for (int i = 0; i < n; i++) //输入排序后的整数

{

printf("%d ", buf[i]);

}

return 0;

}

快速排序

#include <stdio.h>

#define N 7

void qsort (int a[], int left, int right);

int main()

{

int a[10], i;

printf("请输入%d个整数\n", N);

for (i = 0; i < N; i++)

scanf("%d", &a[i]);

qsort(a, 0, N-1);

for (i = 0; i < N; i++)

printf("%d ", a[i]);

return 0;

}

void qsort (int a[], int left, int right)

{

int l, r, value;

if(left >= right)

return;

l = left; r = right;

value = a[l];

do

{

while ((l < r) && (a[r] >= value)) r--;

if(l < r) { a[l] = a[r]; l++; }

while ((l < r) && a[l] <= value) l++;

if(l < r) { a[r] = a[l]; r--; }

} while (l != r);

a[l] = value;

qsort (a, left, l - 1);

qsort (a, r + 1, right);

}

日期计算

数制转换

#include<stdio.h>

#include<string.h>

int main()

{

int a, b; //存放输入进制

char str[40]; //存放输入a进制数

scanf("%d%s%d", &a, str, &b); //输入

int tmp = 0, length = strlen(str), c = 1; //tmp为我们要计算的a进制对应的十进制数，length为字符串长度，c为各个数位的权重初始化为1，表示最低位数位权1，之后的每位权重都是前一位权重的a倍

for (int i = length - 1; i >= 0; i--) { //从低位到高位遍历每一个数位上的数字

int x;

if (str[i] >= '0'&& str[i] <= '9') {

x = str[i] - '0'; //当字符在0到9之间，计算其代表数字

}

else if (str[i] >= 'a'&& str[i] <= 'z') {

x = str[i] - 'a' + 10; //当字符为小写字符时，计算其代表数字

}

else {

x = str[i] - 'A' + 10; //当字符为大写字符时，计算其代表数字

}

tmp += x \* c; //累加该位数字与该数位的权重的积

c \*= a; //计算下一位数权重

}

//转换为b进制

char ans[40], size = 0; //用ans保存转换到b进制的各个数位数字

do {

int x = tmp % b; //计算该位数字

ans[size++] = (x < 10) ? x + '0' : x - 10 + 'A'; //将数字转换为字符（建议改一下）

tmp /= b;

} while (tmp);

for (int i = size - 1; i >= 0; i--) //倒序输出

{

printf("%c", ans[i]);

}

printf("\n");

return 0;

}

素数的埃氏筛

int i, j, a[N+1] = {0};//假设N为常量

a[0] = 1;

a[1] = 1;//0和1不是素数

for (i=2; i<=sqrt(N); i++)

{

if (a[i]!=1)//跳过已被筛掉的数

{

j = 2;

while (i\*j<=N)

{

a[i\*j] = 1;

j++;

}

}

}

for (i=0; i<N; i++)

{

if (!a[i])//如果a[i]==0，则a[i]是素数

{

printf("%d\n", i);

}

}

算法

动态规划：最长递增子序列

#include <iostream>

#include <cstdio>

#include <algorithm>

#include <cstdlib>

#include <cstring>

#include <cmath>

using namespace std;

const int maxn = 103, INF = 0x7f7f7f7f;

int a[maxn], f[maxn];

int n,ans = -INF;

int main()

{

scanf("%d", &n);

for(int i=1; i<=n; i++)

{

scanf("%d", &a[i]);

f[i] = 1;

}

for(int i=1; i<=n; i++)

for(int j=1; j<i; j++)

if(a[j] < a[i])

f[i] = max(f[i], f[j]+1);

for(int i=1; i<=n; i++)

ans = max(ans, f[i]);

printf("%d\n", ans);

return 0;

}

最长公共子序列

for(int i=1;i<=n;i++){

for(int j=1;j<=m;j++)

{

dp[i][j]=max(dp[i-1][j],dp[i][j-1]);

if(A[i]==B[j])

dp[i][j]=max(dp[i][j],dp[i-1][j-1]+1);

}

}

最长回文子序列

#include<iostream>

#include<string>

#include<algorithm>

using namespace std;

//递归方法，求解最长回文子序列

int lps(char \*str, int i, int j)

{

if (i == j)

return 1; //只有一个元素，回文长度为1

if (i > j) return 0; //因为只计算序列str[i....j]

//如果首尾相同

if (str[i] == str[j])

return lps(str, i + 1, j - 1) + 2;

//如果首尾不同

return max(lps(str, i, j - 1), lps(str, i + 1, j));

}

int main()

{

char str[] = "cabbeaf";

int n = strlen(str);

int res = lps(str, 0, n - 1);

cout << res<< endl;

getchar();

return 0;

}

#include<iostream>

#include<algorithm>

using namespace std;

//动态规划求解最长回文子序列，时间复杂度为O(n^2)

int lpsDp(char \*str, int n)

{

int dp[10][10], tmp;

memset(dp, 0, sizeof(dp));

for (int i = 0; i < n; ++i) dp[i][i] = 1;

for (int i = 1; i < n; ++i)

{

tmp = 0;

//考虑所有连续的长度为i+1的子串，str[j....j+i]

for (int j = 0; j + i < n; j++)

{

//如果首尾相同

if (str[j] == str[j + i])

tmp = dp[j + 1][j + i - 1] + 2;

//如果首尾不同

else

tmp = max(dp[j + 1][j + i], dp[j][j + i - 1]);

dp[j][j + i] = tmp;

}

}

return dp[0][n - 1]; //返回字符串str[0...n-1]的最长回文子序列长度

}

int main()

{

char str[10] = "cabbeaf";

int res = lpsDp(str, strlen(str));

cout << res << endl;

getchar();

return 0;

}

素数环代码

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#define MAX\_LENGTH 20

void Primecircle(int i, int\* NumTaken, int\* NumFlag, int\* NA);

int isPrime(int sum);

int n = 0;

int NumTaken[MAX\_LENGTH] = { 0 };

int NumFlag[MAX\_LENGTH] = { 0 };

int main() {

    scanf("%d", &n);

    if (n % 2 == 1) {

        printf("No Answer");//必须是偶数个

        return 0;

    }

    NumFlag[1] = 1;

    NumTaken[1] = 1;

    int NA = 1;

    Primecircle(2, NumTaken, NumFlag, &NA);

    if (NA) printf("No Answer");

    return 0;

}

void Primecircle(int i, int\* NumTaken, int\* NumFlag, int\* NA) {

    int j, k;

    if (i % 2 == 1) j = 3;

    else j = 2;

    for (; j <= n; j += 2) {

        if (NumFlag[j] != 0 || isPrime(NumTaken[i - 1] + j) == 0)

            continue;

        NumTaken[i] = j;

        NumFlag[j] = 1;

        if (i == n) {

            if (isPrime(1 + j)) {

                \*NA = 0;

                for (k = 1; k <= n; k++)

                    printf("%d ", NumTaken[k]);

                printf("\n");

            }

        }

        else {

            Primecircle(i + 1, NumTaken, NumFlag, NA);

        }

        NumTaken[i] = 0;

        NumFlag[j] = 0;

    }

}

int isPrime(int sum) {

    if (sum < 2)

        return 0;

    for (int i = 2; i \* i <= sum; i++) {

        if (sum % i == 0)

            return 0;

    }

    return 1;

}

#define \_CRT\_SECURE\_NO\_DEPRECATE

#include<iostream>

using namespace std;

int n, m, \* pn, \* pm;

scanf("%d",&n);

pn = (int\*)malloc(n \* sizeof(int));

for (int i = 0; i < n; i++)

{

scanf("%d", &pn[i]);

}

scanf("%d", &m);

pm = (int\*)malloc(m \* sizeof(int));

for (int i = 0; i < m; i++)

{

scanf("%d", &pm[i]);

}

scanf("%d", &n);

int\* group = (int\*)malloc(n \* sizeof(int));

for (int i = 0; i < n; i++)

{

scanf("%d", &group[i]);

if (getchar() == ' ')continue;

}

图形用户界面

描述已自动生成

图形用户界面, 文本

描述已自动生成

图形用户界面, 文本

描述已自动生成

以下为一个读取的案例

#include <stdio.h>

#include <iostream>

#include <string.h>

using namespace std;

int main() {

int numofword;

char word[20]; // 容纳更长的单词

int order = 0;

char group[18] = {}; // 存放排序后的结果（每组两位数字）

int flag = 0;

cin >> numofword;

cin.ignore();//忽略缓冲区的一个字符

for (int k = 0; k < numofword; k++) {

// 读取单词

for (int i = 0; i < 20; i++) word[i] = '\0'; // 清空单词缓存

cin >> word;

图形用户界面, 文本, 应用程序

描述已自动生成

文本

描述已自动生成

int main()

{

char ch1, ch2;

for (int i = 0;i < 3;i++) {

scanf("%c %c", &ch1, &ch2);

char a = getchar();//为了吸收多余的回车字符

printf("ch1=%c\n", ch1);

printf("ch2=%c\n", ch2);

}

}

文本

描述已自动生成

图形用户界面, 应用程序

描述已自动生成

以下为指针的实例

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

//以下为课本中的链表样例

//此处注意函数声明，此处省略

struct train\_tag

{

char num[12];

char sender[20];

char name[20];

int weight;

char addr[20];

struct train\_tag\* next;

};

int main() {

//静态链表的创建，假设有10个货物

struct train\_tag array[10] = {};

struct train\_tag \*StaticExampleHead = CreateStatic(array, 10);

//动态链表的创建，假设输入货物重量为0时结束

struct train\_tag\* DynamicExamleHead = CreateDynamic();

//访问链表（以动态非循环链表为例）

Display(StaticExampleHead);

//删除链表节点，以删除目的地为delete\_addr的节点为例

char delete\_addr[10] = {};

scanf("%s", delete\_addr);

Delete(DynamicExamleHead, delete\_addr);

//插入链表节点

struct train\_tag To\_Insert = { "8","清华大学","考试卷子",500,"4吨",NULL };

struct train\_tag\* pNode = &To\_Insert;

Insert(DynamicExamleHead,pNode);

//链表的释放，动态链表最后记得释放内存

Destroy(DynamicExamleHead);

}

struct train\_tag\* CreateStatic(struct train\_tag a[], int n)//数组作为参数传递时，变量类型退化为一个指针

{

struct train\_tag\* head;

int i;

head = &a[0];

for (i = 0; i < n; i++)

{

//输入a[i]各个变量的值，此处忽略

if (i < n - 1)a[i].next = &a[i + 1];//除了尾结点外的next指向下一个

else a[i].next = NULL;//尾结点处理，循环链表时指向a[1]

}

return head;

}

struct train\_tag\* CreateDynamic()//注意在这个过程中，p先行移动到下一对象，next“钩住”下一对象，q再行移动到下一对象

{

int weight;

struct train\_tag\* head=NULL, \* p=NULL, \* q=NULL;

while (1)

{

printf("输入货物重量，0表示结束： ");

scanf("%d", &weight);

if (weight == 0)break;

p = (struct train\_tag\*)malloc(sizeof(struct train\_tag));//申请p的空间

//输入货物的其他信息

if (head == NULL)//首节点

{

head = p;

}

else//后续：q（尚未移动）对应的next为p

{

q->next = p;

}

q = p;//都得让q指向新节点

}

if (head != NULL)q->next = NULL;//循环链表指向第一个节点

return head;

}

void Display(struct train\_tag\* head)

{

struct train\_tag\* p = head;

while (p != NULL)//注：非循环情况下尾结点的next为NULL才适用，在循环链表中注意循环的次数

{

printf("%s", p->name);//此处省略

p = p->next;

}

}

struct train\_tag\* Delete(struct train\_tag \*head, char\* addr)

{

//该函数分为两步：查找与删除；

struct train\_tag\* p, \* q;

if (head == NULL)return NULL;//空链表

p = head;

while ((p != NULL) && strcmp(p->addr, addr))//对比不相等

{

q = p;

p = p->next;

}

if ((p != NULL) && !strcmp(p->addr, addr))//对比相等，删除

{

if (p == head)//删除的是首节点，直接把head后移

head = p->next;

else

q->next = p->next; //在删除时注意p指向的是待删除结点，q指向待删除节点的前一个节点，接着q直接指向p的后一个节点，最后释放p指针指向对象

free(p);

}

return head;

}

struct train\_tag\* Insert(struct train\_tag\* head, struct train\_tag\* pNode)

{

//以下以火车节数按照重量先后顺序排列，例如3 4 6，目标将重量为5的节点插入其中

struct train\_tag\* p, \* q;

if (head == NULL)return pNode;//情况一：空链表

if (pNode->weight <= head->weight)//情况二：新节点货物重量小于首节点

{

pNode->next = head;

head = pNode;

return head;

}

//情况三：新节点货物重量大于首节点

q = head;

p = head->next;

while (p != NULL)//循环链表注意次数

{

if (pNode->weight <= p->weight)break;

else

{

p = p->next;

q = q->next;

}

}

//将pNode插入

pNode->next = p;

q->next = pNode;

return head;

}

void Destroy(struct train\_tag\* head)

{

struct train\_tag\* p, \* q;

p = head;

while (p != NULL)

{

q = p;

p = p->next;

free(q);

}

}

以下为动态二维数组

#include<stdio.h>

#include<stdlib.h>

int main()

{

//以下为动态二维数组的创建，在使用时与普通的二维数组一样

int (\*a)[4], i, j;//这里的4根据题目修改

int nRows = 3, nCols = 4;

//创建动态数组，这里将malloc强制转换为含有 4（题目决定该数） 个 int 的数组的指针

a = (int (\*)[4])malloc(nRows \* nCols \* sizeof(int));

//访问数组

for (i = 0; i < 3; i++)

for (j = 0; j < 4; j++)a[i][j] = 0;

//释放数组

free(a);

}

在行列数为输入时，使用以下方法

#include <stdio.h>

#include <stdlib.h>

int main() {

int n, m;

printf("Enter number of rows and columns: ");

scanf("%d %d", &n, &m);

// 动态分配二维数组

int \*\*a = (int \*\*)malloc(n \* sizeof(int \*));

for (int i = 0; i < n; i++) {

a[i] = (int \*)malloc(m \* sizeof(int));

}

// 输入数组

printf("Enter the matrix elements:\n");

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

scanf("%d", &a[i][j]);

}

}

// 输出数组

printf("The matrix is:\n");

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

printf("%d ", a[i][j]);

}

printf("\n");

}

// 释放内存

for (int i = 0; i < n; i++) {

free(a[i]);

}

free(a);

return 0;

}

核心为

int \*\*a = (int \*\*)malloc(n \* sizeof(int \*));

for (int i = 0; i < n; i++) {

a[i] = (int \*)malloc(m \* sizeof(int));

}

注意二者释放时都需要逐行释放

图形用户界面, 文本

描述已自动生成

图形用户界面, 文本

描述已自动生成

图形用户界面, 文本, 应用程序, 聊天或短信

描述已自动生成

EOF的使用

#include<stdio.h>

int main()

{

int k = 0;

float i, a, b, sum = 0;

while (scanf("%f", &i)!=EOF)

{

a = i;

sum += a;

k++;

}

b = sum / k;

printf("%f\n", b);

system("pause");

return 0;

}

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

#include <stdio.h>

#include <stdlib.h>

int main() {

char \*text = NULL; // 用于存储整段文本

size\_t total\_size = 0; // 总文本大小

char \*line = NULL; // 用于存储每行输入

size\_t len = 0; // 每行缓冲区的长度

ssize\_t read; // 每行读取的字符数

printf("请输入一段文本（以 EOF 结束）：\n");

while ((read = getline(&line, &len, stdin)) != -1) {//代表输入EOF

// 动态扩展总文本的存储空间

text = (char \*)realloc(text, total\_size + read + 1);

if (text == NULL) {

perror("内存分配失败");

free(line);

return 1;

}

// 将新读取的行追加到总文本中

sprintf(text + total\_size, "%s", line);

total\_size += read;

}

// 输出存储的文本

printf("\n你输入的文本是：\n%s", text);

// 释放内存

free(text);

free(line);

return 0;

}

#include <stdio.h>

#include <stdlib.h>

int main() {

char \*text = NULL; // 用于存储整段文本

size\_t size = 0; // 当前分配的内存大小

size\_t len = 0; // 当前文本的长度

int ch; // 当前读取的字符

printf("请输入一段文本（以 EOF 结束）：\n");

while ((ch = getchar()) != EOF) {

// 如果当前内存不足，扩展内存

if (len + 1 >= size) {

size = (size == 0) ? 1 : size \* 2;

text = (char \*)realloc(text, size);

if (text == NULL) {

perror("内存分配失败");

return 1;

}

}

// 将字符追加到文本中

text[len++] = ch;

}

// 添加字符串结束符

text[len] = '\0';

// 输出存储的文本

printf("\n你输入的文本是：\n%s", text);

// 释放内存

free(text);

return 0;

}

输入样例代码

#include<iostream>

#include<string.h>

using namespace std;

int main()

{

char\* text = NULL; // 用于存储整段文本

size\_t size = 0; // 当前分配的内存大小

size\_t len = 0; // 当前文本的长度

char ch; // 当前读取的字符

char now[5] = {};

int i = 0;

printf("请输入一段文本（以 EOF 结束）：\n");

while (1) {

// 如果当前内存不足，扩展内存

if (len + 1 >= size) {

size = (size == 0) ? 1 : size \* 2;

text = (char\*)realloc(text, size);

}

ch = getchar();

// 将字符追加到文本中

text[len++] = ch;

if (i <= 3)

{

now[i] = ch;

i++;

}

if (ch == '\n')

{

i = 0;

now[4] = '\0';

if (strcmp(now, "EOF\n") == 0)break;

}

}

// 添加字符串结束符

text[len] = '\0';

// 输出存储的文本

printf("\n你输入的文本是：\n%s", text);

// 释放内存

free(text);

return 0;

}

类似的代码片段

while (1) {

// 如果当前内存不足，扩展内存

if (len + 1 >= size) {

size = (size == 0) ? 1 : size \* 2;

text = (char\*)realloc(text, size);

}

ch = getchar();

// 将字符追加到文本中

text[len++] = ch;

if (i <= 3)

{

now[i] = ch;

i++;

}

if (ch == '\n')

{

i = 0;

now[4] = '\0';

if (strcmp(now, "EOF\n") == 0)break;

}

}

// 添加字符串结束符

text[len] = '\0';

大小写转化

#include <stdio.h>

#include <ctype.h>

int main() {

char ch;

printf("请输入一个字母:");

scanf("%c", &ch);

// 小写转大写

ch = toupper(ch);

printf("转换为大写: %c\n", ch);

// 大写转小写

ch = tolower(ch);

printf("转换为小写: %c\n", ch);

return 0;

}

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

#include <stdio.h>

#include <iostream>

#include <string.h>

using namespace std;

int main() {

    int numofword;

    char word[20];  // 容纳更长的单词

    int order = 0;

    char group[18] = {}; // 存放排序后的结果（每组两位数字）

    int flag = 0;

    cin >> numofword;

    cin.ignore();//手动清空缓冲区

    for (int k = 0; k < numofword; k++) {

        // 读取单词

        for (int i = 0; i < 20; i++) word[i] = '\0'; // 清空单词缓存

        cin >> word;

        // 转换为小写字母

        for (int i = 0; word[i] != '\0'; i++) {

            if (word[i] >= 'A' && word[i] <= 'Z') {

                word[i] += ('a' - 'A');

            }

        }

        // 数字单词判断并计算平方值取模

        if (strcmp(word, "zero") == 0) {

            group[2 \* order] = '0';

            group[2 \* order + 1] = '0';

            order++;

        }

        else if (strcmp(word, "one") == 0) {

            group[2 \* order] = '0';

            group[2 \* order + 1] = '1';

            order++;

        }

        else if (strcmp(word, "two") == 0) {

            group[2 \* order] = '0';

            group[2 \* order + 1] = '4';

            order++;

        }

        else if (strcmp(word, "three") == 0) {

            group[2 \* order] = '0';

            group[2 \* order + 1] = '9';

            order++;

        }

        else if (strcmp(word, "four") == 0) {

            group[2 \* order] = '1';

            group[2 \* order + 1] = '6';

            order++;

        }

        else if (strcmp(word, "five") == 0) {

            group[2 \* order] = '2';

            group[2 \* order + 1] = '5';

            order++;

        }

        else if (strcmp(word, "six") == 0) {

            group[2 \* order] = '3';

            group[2 \* order + 1] = '6';

            order++;

        }

        else if (strcmp(word, "seven") == 0) {

            group[2 \* order] = '4';

            group[2 \* order + 1] = '9';

            order++;

        }

        else if (strcmp(word, "eight") == 0) {

            group[2 \* order] = '6';

            group[2 \* order + 1] = '4';

            order++;

        }

        else if (strcmp(word, "nine") == 0) {

            group[2 \* order] = '8';

            group[2 \* order + 1] = '1';

            order++;

        }

        else if (strcmp(word, "ten") == 0) {

            group[2 \* order] = '0';

            group[2 \* order + 1] = '0';

            order++;

        }

        else if (strcmp(word, "eleven") == 0) {

            group[2 \* order] = '2';

            group[2 \* order + 1] = '1';

            order++;

        }

        else if (strcmp(word, "twelve") == 0) {

            group[2 \* order] = '4';

            group[2 \* order + 1] = '4';

            order++;

        }

        else if (strcmp(word, "thirteen") == 0) {

            group[2 \* order] = '6';

            group[2 \* order + 1] = '9';

            order++;

        }

        else if (strcmp(word, "fourteen") == 0) {

            group[2 \* order] = '9';

            group[2 \* order + 1] = '6';

            order++;

        }

        else if (strcmp(word, "fifteen") == 0) {

            group[2 \* order] = '2';

            group[2 \* order + 1] = '5';

            order++;

        }

    }

    // 冒泡排序

    for (int i = 0; i < order - 1; i++) {

        for (int j = 0; j < order - 1 - i; j++) {

            if (group[2 \* j] > group[2 \* (j + 1)] ||

                (group[2 \* j] == group[2 \* (j + 1)] && group[2 \* j + 1] > group[2 \* (j + 1) + 1])) {

                // 交换高位

                char temp = group[2 \* j];

                group[2 \* j] = group[2 \* (j + 1)];

                group[2 \* (j + 1)] = temp;

                // 交换低位

                temp = group[2 \* j + 1];

                group[2 \* j + 1] = group[2 \* (j + 1) + 1];

                group[2 \* (j + 1) + 1] = temp;

            }

        }

    }

    int clear = 1;

    for (int i = 0; i < 2 \* order; i++) {

        if (group[i] == '0' && clear) {

            if (i == 2 \* order - 1) cout << '0';

            continue;

        }

        if (group[i] != '0') clear = 0;

        cout << group[i];

    }

    cout << endl;

    return 0;

}

图形用户界面

描述已自动生成

文本

描述已自动生成

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<iostream>

using namespace std;

void ADD(int\* num, struct DNA\* group);

void SELECT(int\* num, struct DNA\* group);

void CUT(int\* num, struct DNA\* group);

void ORDER(int\* num, struct DNA\* group);

struct DNA {

    char type[3];

};

int main() {

    int op, num = 0;

    struct DNA group[1000];

    cin >> op; // 输入操作数

    for (int k = 0; k < op; k++) {

        char todo[10];

        cin >> todo;

        if (!strcmp(todo, "ADD")) ADD(&num, group);

        else if (!strcmp(todo, "SELECT")) SELECT(&num, group);

        else if (!strcmp(todo, "CUT")) CUT(&num, group);

        else if (!strcmp(todo, "ORDER")) ORDER(&num, group);

        // 特定规则检查并替换 DNA 类型

    }

    for (int k = 0; k < num; k++) {

        cout << group[k].type << endl;

    }

    return 0;

}

void ADD(int\* num, struct DNA\* group) {

    cin >> group[\*num].type;

    (\*num)++;

    int q = \*num - 3;

    if (\*num >= 3) {

        if ((!strcmp(group[q].type, "TA")) && (!strcmp(group[q + 1].type, "TA")) && (!strcmp(group[q + 2].type, "TA"))) {

            strcpy(group[q].type, "CG");

            strcpy(group[q + 1].type, "CG");

            strcpy(group[q + 2].type, "CG");

        }

        if (((!strcmp(group[q].type, "TA")) && (!strcmp(group[q + 1].type, "AT")) && (!strcmp(group[q + 2].type, "GC"))) ||

            ((!strcmp(group[q].type, "AT")) && (!strcmp(group[q + 1].type, "TA")) && (!strcmp(group[q + 2].type, "GC"))) ||

            ((!strcmp(group[q].type, "GC")) && (!strcmp(group[q + 1].type, "AT")) && (!strcmp(group[q + 2].type, "TA"))) ||

            ((!strcmp(group[q].type, "AT")) && (!strcmp(group[q + 1].type, "GC")) && (!strcmp(group[q + 2].type, "TA"))) ||

            ((!strcmp(group[q].type, "GC")) && (!strcmp(group[q + 1].type, "TA")) && (!strcmp(group[q + 2].type, "AT"))) ||

            ((!strcmp(group[q].type, "TA")) && (!strcmp(group[q + 1].type, "GC")) && (!strcmp(group[q + 2].type, "AT")))) {

            strcpy(group[q].type, "GC");

            strcpy(group[q + 1].type, "GC");

            strcpy(group[q + 2].type, "GC");

        }

    }

}

void SELECT(int\* num, struct DNA\* group) {

    int index;

    char change[3];

    cin >> index >> change;

    if (index > 0 && index <= \*num) {

        strcpy(group[index - 1].type, change);

    }

}

void CUT(int\* num, struct DNA\* group) {

    int index;

    cin >> index;

    if (index > 0 && index <= \*num) {

        for (int i = index; i < \*num; i++) {

            strcpy(group[i - 1].type, group[i].type);

        }

        (\*num)--;

    }

}

void ORDER(int\* num, struct DNA\* group) {

    int start, endnum;

    cin >> start >> endnum;

    if (start > 0 && endnum <= \*num && start < endnum) {

        for (int i = start - 1; i < endnum - 1; i++) {

            for (int j = start - 1; j < endnum - 1 - (i - start + 1); j++) {

                if (strcmp(group[j].type, group[j + 1].type) > 0) {

                    char temp[3];

                    strcpy(temp, group[j].type);

                    strcpy(group[j].type, group[j + 1].type);

                    strcpy(group[j + 1].type, temp);

                }

            }

        }

    }

}

图形用户界面

描述已自动生成