Experiment13-董皓彧

环境：

```
gcc.exe (x86_64-win32-seh-rev1, Built by MinGW-Builds project) 13.2.0
Visual Stdio Code 1.84.2
```

作业仓库地址：
https://github.com/FHYQ-Dong/Tsinghua-Program-Design-Assignments/tree/main/Experiment13

必做题

Experiment13-1

题目：

```
学生信息统计查询（学号、姓名、性别、出生日期、分数）
```

输入格式：

```
命令行工具，输入 ./executable.exe --help 查看用法
```

输出格式：

```
略
```

代码：

见 Github Repo 或 Tsinghua Git Repo

可执行文件：

见 Github Release 或 Tsinghua Git Release

Experiment13-2

题目：

```
使用枚举类型编写程序，实现输入 1-7，输出对应的英文星期单词
```

输入格式：

```
一个数字，1-7
```

输出格式：

```
一个单词，对应的英文星期单词
```

代码：

```c
#include <stdio.h>
typedef enum {Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday}
Weekday;
char *weekdayName[] = {"Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday", "Sunday"};

int main() {
    Weekday oneDay; scanf("%d", &oneDay);
    printf("%s\n", weekdayName[oneDay-1]);
    return 0;
}
```

输入1：

```
1
```

输出1：

```
Monday
```

输入2：

```
2
```

输出2：

```
Tuesday
```

输入3：

```
3
```

输出3：

```
Wednesday
```

输入4：

```
4
```

输出4：

```
Thursday
```

输入5：

```
5
```

输出5：

```
Friday
```

输入6：

```
6
```

输出6：

```
Saturday
```

输入7：

```
7
```

输出7：

```
Sunday
```

Experiment13-3

题目：

定义结构体 struct Fraction，包含两个整型成员变量 numerator 和 denominator，分别表示分子和分母。编写程序，实现分数的加法运算，并计算以下数列的和，其中 n=10

$$4 \times [1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \cdots + \frac{(-1)^{n-1}}{2n-1}]$$

输入格式：

```
无
```

输出格式：

```
一行，两个数，为数列的和的不同表示方法，用空格隔开：
    第一个：xxx/xxx 形式的最简分数；
    第二个：小数形式的结果
```

代码：

```c
#include <stdio.h>

typedef struct {
    long long numerator;
    unsigned long long denominator;
} Fraction;

long long lgcd(long long a, long long b) {
    long long t;
    while (b) {
        t = a % b;
```

```
        a = b;
        b = t;
    }
    return a;
}
Fraction Fra_normalize(Fraction a) {
    if (a.denominator == 0) {
        a.numerator = 0;
        return a;
    }
    long long gcd;
    gcd = lgcd(a.numerator, a.denominator);
    a.numerator /= gcd;
    a.denominator /= gcd;
    return a;
}
Fraction Fra_add(Fraction a, Fraction b) {
    if (a.denominator == 0 || b.denominator == 0) {
        Fraction c;
        c.numerator = 0;
        c.denominator = 0;
        return c;
    }
    Fraction c;
    long long gcd;
    c.numerator = a.numerator * b.denominator + b.numerator * a.denominator;
    c.denominator = a.denominator * b.denominator;
    return Fra_normalize(c);
}
Fraction Fra_sub(Fraction a, Fraction b) {
    if (a.denominator == 0 || b.denominator == 0) {
        Fraction c;
        c.numerator = 0;
        c.denominator = 0;
        return c;
    }
    Fraction c;
    long long gcd;
    c.numerator = a.numerator * b.denominator - b.numerator * a.denominator;
    c.denominator = a.denominator * b.denominator;
    return Fra_normalize(c);
}
double Fra_to_Double(Fraction a) {
    return (double)a.numerator / a.denominator;
}
int main() {
    Fraction ans = {0, 1}, tmp;
    for (int i=1; i<=10; ++i) {
        tmp.numerator = (i%2 ? 1 : -1);
        tmp.denominator = 2*i - 1;
        ans = Fra_add(ans, tmp);
    }
    ans.numerator *= 4;
    ans = Fra_normalize(ans);
    printf("%lld/%lld %lf", ans.numerator, ans.denominator, Fra_to_Double(ans));
```

```
    return 0;
}
```

输入1：

```

```

输出1：

```
44257352/14549535 3.041840
```

选做题

Optional-Experiment13-1

题目：

```
定义一个双向链表，实现插入和按下标删除的函数
```

输入格式：

```
共 n+m+2 行：
    第一行：一个整数 n，表示插入的元素个数；
    接下来 n 行，每行两个整数，表示插入的元素的下标和值；
    接下来一行，一个整数 m，表示删除的元素个数；
    接下来 m 行，每行一个整数，表示删除的元素的下标
```

输出格式：

```
共 n+m 行：
    第一行：n 个 (index, value) 形式的元组，表示插入后的链表；
    接下来 m 行，每行表示删除一个元素后的链表
```

代码：

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

typedef struct tagNode {
    int index, value;
    struct tagNode *prev;
    struct tagNode *next;
} Node, *PNode;
typedef struct tagList {
    PNode head;
    int length;
} List;

List insert(List *list, int index, int value) {
    PNode p = list->head;
    while (p->next != NULL && p->next->index < index) p = p->next;
```

```
    if (p->next != NULL && p->next->index == index) {
        p->next->value = value;
        return *list;
    }
    PNode q = (PNode)malloc(sizeof(Node));
    memset(q, 0, sizeof(Node));
    q->index = index;
    q->value = value;
    q->prev = p;
    q->next = p->next;
    p->next = q;
    if (q->next != NULL) q->next->prev = q;
    return *list;
}

void delete_by_index(List *list, int index) {
    PNode p = list->head;
    while (p->next != NULL && p->next->index < index) p = p->next;
    if (p->next == NULL || p->next->index > index) return;
    PNode q = p->next;
    p->next = q->next;
    if (q->next != NULL) q->next->prev = p;
    free(q);
}

List create_list() {
    List list;
    PNode p = (PNode)malloc(sizeof(Node));
    memset(p, 0, sizeof(Node));
    list.head = p;
    list.length = 0;
    return list;
}

void print_list(List *list) {
    PNode p = list->head->next;
    while (p != NULL) {
        printf("(%d, %d) ", p->index, p->value);
        p = p->next;
    }
    printf("\n");
}

int main() {
    int n; scanf("%d", &n);
    List list = create_list();
    for (int i=1; i<=n; ++i) {
        int idx, val;
        scanf("%d%d", &idx, &val);
        insert(&list, idx, val);
    }
    print_list(&list);
    int m; scanf("%d", &m);
    for (int i=1; i<=m; ++i) {
        int idx; scanf("%d", &idx);
```

```
        delete_by_index(&list, idx);
        print_list(&list);
    }
    return 0;
}
```

输入1：

```
5
0 1
1 2
2 3
3 4
4 5
3
0
2
4
```

输出1：

```
(0, 1) (1, 2) (2, 3) (3, 4) (4, 5)
(1, 2) (2, 3) (3, 4) (4, 5)
(1, 2) (3, 4) (4, 5)
(1, 2) (3, 4)
```

输入2：

```
10
0 1
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9
9 10
5
0
2
4
6
8
```

输出2：

```
(0, 1) (1, 2) (2, 3) (3, 4) (4, 5) (5, 6) (6, 7) (7, 8) (8, 9) (9, 10)
(1, 2) (2, 3) (3, 4) (4, 5) (5, 6) (6, 7) (7, 8) (8, 9) (9, 10)
(1, 2) (3, 4) (4, 5) (5, 6) (6, 7) (7, 8) (8, 9) (9, 10)
(1, 2) (3, 4) (5, 6) (6, 7) (7, 8) (8, 9) (9, 10)
(1, 2) (3, 4) (5, 6) (7, 8) (8, 9) (9, 10)
(1, 2) (3, 4) (5, 6) (7, 8) (9, 10)
```

Optional-Experiment13-2

题目：

> 定义一个链表，其元素的值均为正整数。反复找出链表中最小的元素，将其从链表中删除，直到链表为空为止。输出每次删除的元素值。

输入格式：

```
共 2 行：
    第一行：一个整数 n，表示链表的元素个数；
    第二行：n 个整数，用空格隔开，表示链表的元素值
```

输出格式：

```
共 n 个整数，用空格隔开，表示每次删除的元素值
```

代码：

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int value;
    struct Node *next;
} Node;

void insert(Node *head, int value) {
    Node *p = head;
    while (p->next != NULL) p = p->next;
    p->next = (Node *)malloc(sizeof(Node));
    p->next->next = NULL;
    p->next->value = value;
}

void del(Node *head, int value) {
    Node *p = head;
    while (p->next != NULL) {
        if (p->next->value == value) {
            Node *q = p->next;
            p->next = p->next->next;
            free(q);
            break;
        }
        p = p->next;
    }
```

```
}

int get_min(Node *head) {
    Node *p = head;
    int min = 0x7fffffff;
    while (p->next != NULL) {
        if (p->next->value < min) min = p->next->value;
        p = p->next;
    }
    printf("%d ", min);
    return min;
}

int main() {
    int n; scanf("%d", &n);
    Node *head = (Node *)malloc(sizeof(Node));
    head->next = NULL; head->value = -1;
    for (int i=1; i<=n; ++i) {
        int val; scanf("%d", &val);
        insert(head, val);
    }
    while (head->next != NULL) {
        int min = get_min(head);
        del(head, min);
    }
    return 0;
}
```

输入1：

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

输出1：

```
1 2 3 4 5
```

输入2：

```
10
6239 303 121 5 93023 40 235 97 875 124
```

输出2：

```
5 40 97 121 124 235 303 875 6239 93023
```

Optional-Experiment13-3

题目：

```
执行下列程序的输出结果是?
```

输入格式：

```
无
```

输出格式：

```
程序的输出结果（具体解析见注释）
```

代码：

```
#include <stdio.h>

typedef struct {
    char name[9];
    char sex;
    int score[3];
} STU;

void f(STU *a) { // a 是指针，函数内可以直接修改 main() 中变量的值
    STU b = {"huang", 'm', 81, 92}, *p = &b;
    *a = *p;
    a->sex = 'f';
    a->score[2] = a->score[0] + a->score[1];
}

int main() {
    STU c = {"Qian", 'f', 93, 97}, *d = &c;
    f(&c); // 无论 c 是什么，统一修改为 void f() 中的 b.
    printf("%s, %c, %d, %d, %d\n", d->name, d->sex, d->score[0], d->score[1], d-
>score[2]);
    return 0;
}
```

输入1：

```

```

输出1：

```
huang, f, 81, 92, 173
```

Optional-Experiment13-4

题目：

```
使用循环链表解决约瑟夫问题，即 n 个人围成一圈，从第一个人开始报数，报到 3 的人出列，再由下一个人重新从 1 开始报数，报到 3 的人出列，如此循环，直到剩余 2 人，输出最后剩余的两个人的编号。从 1 开始编号。
```

输入格式：

一个整数，总人数

输出格式：

两个整数，用空格隔开，最后剩余的两个人的编号

代码：

```
#include <stdio.h>
#include <stdlib.h>

typedef struct Node {
    int idx;
    struct Node *next;
} Node;

int main() {
    int n; scanf("%d", &n);
    Node *head = (Node *)malloc(sizeof(Node)), *cur;
    head->idx = 1; head->next = NULL; cur = head;
    for (int i=2; i<=n; ++i) {
        Node *p = (Node *)malloc(sizeof(Node));
        p->idx = i; p->next = NULL;
        cur->next = p; cur = p;
    }
    cur->next = head; cur = head;
    for (int i=1; i<=n-2; ++i) {
        cur = cur->next;
        Node *tmp = cur->next;
        cur->next = tmp->next;
        free(tmp);
        cur = cur->next;
    }
    printf("%d %d\n", cur->idx, cur->next->idx);
    return 0;
}
```

输入1：

```
5
```

输出1：

```
2 4
```

输入2：

```
10
```

输出2：

```
10 4
```

输入3：

```
100
```

输出3：

```
58 91
```

输入4：

```
1000
```

输出4：

```
226 604
```

输入5：

```
10000
```

输出5：

```
8923 2692
```