

基础实验：实现文件读写与向量

姓名：黎行健

学号：2024201585

2025 年 10 月 22 日

1 问题分析

文件的读写要用到文件 IO 流，向量的功能实现则可以利用普通顺序表的动态分配内存以及基础操作完成

2 数据结构设计与实现

实现 Myarray 结构体，其对象即为向量

2.1 ADT 设计

- 数据对象：Myarray 结构体，内含 size（向量长度）arr（数组，向量元素存储空间），数组 arr 内每个元素都是 int 类型

```
typedef struct {  
    int size;  
    int *arr;  
} Myarray;
```

- 数据关系：arr 是顺序的，每个元素除了头元素和尾元素都有唯一的前驱和后继元素，头元素有唯一的后继元素但没有前驱元素，尾元素有唯一的前驱元素但没有后继元素
- 基本操作：

```
setsize(x): 设置 size  
getsize(): 获取 size  
initial_arr(p): 初始化向量  
show(): 打印向量  
push_back_x(x): 在向量末尾添加元素  
pop_back(): 在向量末尾弹出元素  
insert_element(pos,x): 在向量pos位处插入元素  
erase_element(pos): 删除pos位置的元素  
update_element(pos,x): 修改pos位置的元素  
save_array(filename,length): 保存向量数据到指定文件里面
```

2.2 ADT 的物理实现

```
typedef struct {
    int size;
    int *arr;
} Myarray;
void getsize(Myarray *my);
void setsize(Myarray *my, int num);
void initial_arr(Myarray *my, int *p);
void show(Myarray *my);
void push_back_x(Myarray *my, int x);
void pop_back(Myarray *my);
void insert_element(Myarray *my, int pos, int x);
void erase_element(Myarray *my, int pos);
void update_element(Myarray *my, int pos, int x);
void save_array(Myarray *my, char *filename, int length);
```

3 算法设计与实现

3.1 算法设计及实现

整个流程，在选择文件输入之后，我选择了循环结构，支持用户不断进行对向量的操作，直到用户发出 exit 指令

3.1.1 文件写入写出

根据不同类型，进行不同的读和写操作其中，读入文件的时候用了额外的数组

- 时间复杂度分析: $O(n)$
- 空间复杂度分析: $O(n)$

这里只展示输出文件 save 的实现，先判断文件类型，再对应输出

```
void save_array(Myarray *my, char *filename, int length) {
    if (my->arr == NULL) {
        printf("Error!!! You have not initialized your array yet!\n");
        ;
        return;
    }
    char root[] = "./output/";
    int root_len = my_strlen(root);
    char *fullpath = (char*)malloc(root_len + length + 1);
```

```

// 手动构建完整路径
my_strcpy(fullpath, root);
my_strcat(fullpath, filename);

if (filename[length - 1] == 't') {
    FILE *fout = fopen(fullpath, "w");
    if (fout == NULL) {
        printf("Error opening file for writing!\n");
        free(fullpath);
        return;
    }
    fprintf(fout, "%d\n", my->size);
    for (int i = 0; i < my->size; i++) {
        fprintf(fout, "%d", my->arr[i]);
        if (i < my->size - 1)
            fprintf(fout, " ");
    }
    fclose(fout);
}
else {
    FILE *fout = fopen(fullpath, "wb");
    if (fout == NULL) {
        printf("Error opening file for writing!\n");
        free(fullpath);
        return;
    }
    fwrite(&my->size, sizeof(int), 1, fout);
    fwrite("\n", sizeof(char), 1, fout);
    for (int i = 0; i < my->size; i++) {
        fwrite(&my->arr[i], sizeof(int), 1, fout);
        if (i < my->size - 1)
            fwrite(" ", sizeof(char), 1, fout);
    }
    fclose(fout);
}
free(fullpath);
}

```

3.1.2 输入处理

共 9 种输入格式，分别对应题目要求的八个操作和退出，利用 funcname、op1、op2 三个字符数组存储对应的函数名、参数 1 和参数 2

- 时间复杂度分析: $O(n)$
- 空间复杂度分析: $O(n)$

```

char instruction[1000];
printf("请输入: \n");
if (loop_count == 0) {
    getchar(); // 清除缓冲区
}
fgets(instruction, 1000, stdin);

// 使用自定义函数移除换行符
remove_newline(instruction);

char func_name[1000] = {0}, op1[1000] = {0}, op2[1000] = {0};
int id1 = 0, id2 = 0, id3 = 0;
int face_blank = 0;

for (int i = 0; instruction[i] != '\0'; i++) {
    if (instruction[i] == ' ') {
        face_blank++;
        continue;
    }
    switch (face_blank) {
        case 0: {
            func_name[id1] = instruction[i];
            id1++;
            break;
        }
        case 1: {
            op1[id2] = instruction[i];
            id2++;
            break;
        }
        case 2: {
            op2[id3] = instruction[i];
            id3++;
            break;
        }
    }
}

switch (matching_mode(func_name, id1)) {
    case 1: {

```

```

        getsize(&my);
        break;
    }
    case 2: {
        show(&my);
        break;
    }
    case 3: {
        int val = charToint(op1, id2);
        push_back_x(&my, val);
        break;
    }
    case 4: {
        pop_back(&my);
        break;
    }
    case 5: {
        int pos = charToint(op1, id2);
        int val = charToint(op2, id3);
        insert_element(&my, pos, val);
        break;
    }
    case 6: {
        int pos = charToint(op1, id2);
        erase_element(&my, pos);
        break;
    }
    case 7: {
        int pos = charToint(op1, id2);
        int val = charToint(op2, id3);
        update_element(&my, pos, val);
        break;
    }
    case 8: {
        save_array(&my, op1, id2);
        break;
    }
    case 9: {
        printf("程序运行结束\n");
        free_array(&my);
        return 0;
    }
    default: {

```

```

        printf("错误, 请重新输入\n");
    }
}
printf("完成了一次操作!按Enter键继续...");
getchar();
system("clear");
loop_count++;
}

```

3.1.3 函数名匹配

利用自定义 cmp 函数, 返回对应 case 值, 进入对应 switch 分支, 执行对应函数

- 时间复杂度分析: $O(n)$
- 空间复杂度分析: $O(n)$

实现函数匹配

```

// 字符串比较函数
int cmp(char* string1, char* string2, int length) {
    for (int i = 0; i < length; i++) {
        if (string1[i] != string2[i]) return 0;
    }
    return 1;
}

// 命令匹配函数
int matching_mode(char *func_name, int size) {
    if (size == 4) {
        char func1[] = "size";
        char func2[] = "show";
        char func3[] = "save";
        char func4[] = "exit";
        if (cmp(func_name, func1, 4)) return 1;
        else if (cmp(func_name, func2, 4)) return 2;
        else if (cmp(func_name, func3, 4)) return 8;
        else if (cmp(func_name, func4, 4)) return 9;
        else return -1;
    }
    else if (size == 5) {
        char func[] = "erase";
        if (cmp(func_name, func, 5)) return 6;
        else return -1;
    }
}

```

```

    }
    else if (size == 6) {
        char func1[] = "insert";
        char func2[] = "update";
        if (cmp(func_name, func1, 6)) return 5;
        else if (cmp(func_name, func2, 6)) return 7;
        else return -1;
    }
    else if (size == 8) {
        char func[] = "pop_back";
        if (cmp(func_name, func, 8)) return 4;
        else return -1;
    }
    else if (size == 9) {
        char func[] = "push_back";
        if (cmp(func_name, func, 9)) return 3;
        else return -1;
    }
    else {
        return -1;
    }
}

```

3.1.4 向量初始化

将外部读入的数组存入向量

- 时间复杂度分析: $O(n)$
- 空间复杂度分析: $O(n)$

3.1.5 打印向量

- 时间复杂度分析: $O(n)$
- 空间复杂度分析: $O(n)$

3.1.6 向量尾部进行 push

需要重新分配动态内存

- 时间复杂度分析: $O(1)$
- 空间复杂度分析: $O(n)$

```

void push_back_x(Myarray *my, int x) {
    if (my->arr == NULL) {
        printf("Error!!! You have not initialized your array yet!\n");
        ;
        return;
    }
    int *newarr = (int*)malloc((my->size + 1) * sizeof(int));
    for (int i = 0; i < my->size; i++) {
        newarr[i] = my->arr[i];
    }
    newarr[my->size] = x;
    my->size++;
    free(my->arr);
    my->arr = newarr;
    show(my);
}

```

3.1.7 插入数据进向量某个位置

使得 pos 位置后所有元素都向后移动一位

- 时间复杂度分析: $O(n)$
- 空间复杂度分析: $O(n)$

```

void insert_element(Myarray *my, int pos, int x) {
    if (my->arr == NULL) {
        printf("Error!!! You have not initialized your array yet!\n");
        ;
        return;
    }
    if (pos <= 0 || pos > my->size + 1) {
        printf("Invalid index!!! please try another...\n");
        return;
    }
    int *newarr = (int*)malloc((my->size + 1) * sizeof(int));
    for (int i = 0; i < pos - 1; i++) {
        newarr[i] = my->arr[i];
    }
    for (int i = my->size; i > pos - 1; i--) {
        newarr[i] = my->arr[i - 1];
    }
    newarr[pos - 1] = x;
}

```



```

    my->size++;
    free(my->arr);
    my->arr = newarr;
    show(my);
}

```

3.1.8 向量尾部 pop

- 时间复杂度分析: $O(1)$
- 空间复杂度分析: $O(1)$

3.1.9 向量内部删除元素

pos 位置之后元素都要向前移动一位

- 时间复杂度分析: $O(n)$
- 空间复杂度分析: $O(1)$

```

void erase_element(Myarray *my, int pos) {
    if (my->arr == NULL) {
        printf("Error!!! You have not initialized your array yet!\n");
        ;
        return;
    }
    if (pos <= 0 || pos > my->size) {
        printf("Invalid index!!! please try another...\n");
        return;
    }
    for (int i = pos; i < my->size; i++) {
        my->arr[i - 1] = my->arr[i];
    }
    my->size--;
    show(my);
}

```

3.1.10 更新 pos 元素

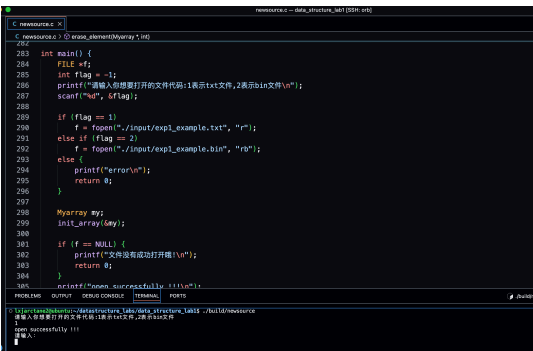
- 时间复杂度分析: $O(1)$
- 空间复杂度分析: $O(1)$

4 实验环境

- 硬件环境：Macos M1 Pro、内存 16GB
- 软件环境：操作系统：Ubuntu 22.04 IDE：vscode language：c

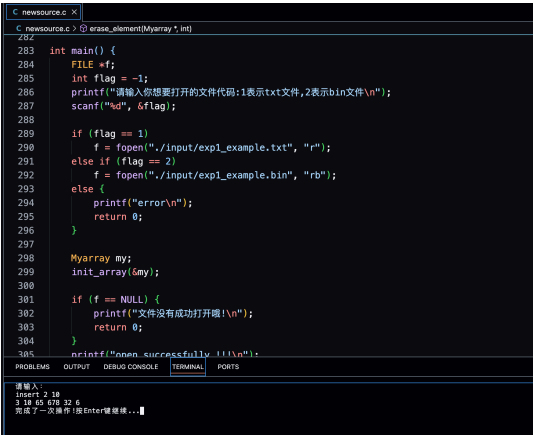
5 实验结果与分析

在每次输入之后，都能够正确的完成对应的操作，展示图片如下：



```
C newsource.c
283 int main() {
284     FILE *f;
285     int flag = -1;
286     printf("请输入想要打开的文件代码:1表示txt文件,2表示bin文件\n");
287     scanf("%d", &flag);
288
289     if (flag == 1)
290         f = fopen("./input/exp1_example.txt", "r");
291     else if (flag == 2)
292         f = fopen("./input/exp1_example.bin", "rb");
293     else {
294         printf("error\n");
295         return 0;
296     }
297
298     Myarray my;
299     init_array(&my);
300
301     if (f == NULL) {
302         printf("文件没有成功打开\n");
303         return 0;
304     }
305
306     printf("操作成功\n");
307
308     return 0;
309 }
```

图 1: open file



```
C newsource.c
283 int main() {
284     FILE *f;
285     int flag = -1;
286     printf("请输入想要打开的文件代码:1表示txt文件,2表示bin文件\n");
287     scanf("%d", &flag);
288
289     if (flag == 1)
290         f = fopen("./input/exp1_example.txt", "r");
291     else if (flag == 2)
292         f = fopen("./input/exp1_example.bin", "rb");
293     else {
294         printf("error\n");
295         return 0;
296     }
297
298     Myarray my;
299     init_array(&my);
300
301     if (f == NULL) {
302         printf("文件没有成功打开\n");
303         return 0;
304     }
305
306     printf("操作成功\n");
307
308     return 0;
309 }
```

图 2: insert

```
C:\newsource.c X
C:\newsource.c> @ erase_element(Myarray, 10)
282
283 int main() {
284     FILE *f;
285     int flag = -1;
286     printf("请输入你想要打开的文件代码:1表示txt文件,2表示bin文件\n");
287     scanf("%d", &flag);
288
289     if (flag == 1)
290         f = fopen("./input/exp1_example.txt", "r");
291     else if (flag == 2)
292         f = fopen("./input/exp1_example.bin", "rb");
293     else {
294         printf("error\n");
295         return 0;
296     }
297
298     Myarray my;
299     init_array(&my);
300
301     if (f == NULL) {
302         printf("文件没有成功打开哦!\n");
303         return 0;
304     }
305     printf("open successfully !!!\n");
306
307 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
请输入:
3 18 45 68 32
完成了一次操作!按Enter键继续...
```

图 3: pop_{back}

```
C:\newsource.c X
C:\newsource.c> @ erase_element(Myarray, 10)
282
283 int main() {
284     FILE *f;
285     int flag = -1;
286     printf("请输入你想要打开的文件代码:1表示txt文件,2表示bin文件\n");
287     scanf("%d", &flag);
288
289     if (flag == 1)
290         f = fopen("./input/exp1_example.txt", "r");
291     else if (flag == 2)
292         f = fopen("./input/exp1_example.bin", "rb");
293     else {
294         printf("error\n");
295         return 0;
296     }
297
298     Myarray my;
299     init_array(&my);
300
301     if (f == NULL) {
302         printf("文件没有成功打开哦!\n");
303         return 0;
304     }
305     printf("open successfully !!!\n");
306
307 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
请输入:
erase 2
3 18 45 68 32
完成了一次操作!按Enter键继续...
```

图 4: erase

```
EXPLORER C:\newsource.c X
OPEN EDITORS
C:\newsource.c
DATA_STRUCTURE_LAB1.c
> ascode
> build
> input
> output
> exp1_bin
> exp1_binout
C:\newsource.c
C:\source.cpp
C:\newsource.c X
C:\newsource.c> @ erase_element(Myarray, 10)
282
283 int main() {
284     FILE *f;
285     int flag = -1;
286     printf("请输入你想要打开的文件代码:1表示txt文件,2表示bin文件\n");
287     scanf("%d", &flag);
288
289     if (flag == 1)
290         f = fopen("./input/exp1_example.txt", "r");
291     else if (flag == 2)
292         f = fopen("./input/exp1_example.bin", "rb");
293     else {
294         printf("error\n");
295         return 0;
296     }
297
298     Myarray my;
299     init_array(&my);
300
301     if (f == NULL) {
302         printf("文件没有成功打开哦!\n");
303         return 0;
304     }
305     printf("open successfully !!!\n");
306
307 PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
请输入:
exp1_bin
完成了一次操作!按Enter键继续...
```

图 5: save as bin

```
C:\newsources> gcc erase_element(Myarray.c)

283 int main() {
284     FILE *f;
285     int flag = -1;
286     printf("请输入您想要打开的文件代码:1表示txt文件,2表示bin文件\n");
287     scanf("%d", &flag);
288
289     if (flag == 1)
290         f = fopen("./input/exp1_example.txt", "r");
291     else if (flag == 2)
292         f = fopen("./input/exp1_example.bin", "rb");
293     else {
294         printf("error\n");
295         return 0;
296     }
297
298     Myarray my;
299     init_array(my);
300
301     if (f == NULL) {
302         printf("文件没有成功打开\n");
303         return 0;
304     }
305     printf("open successfully !!!\n");
306 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

请输入 0 1
程序运行结果
C:\newsources> gcc erase_element(Myarray.c)
请输入您想要打开的文件代码:1表示txt文件,2表示bin文件
2
open successfully !!!
请输入:
0 1
3 65 678 32 6
您输入了一次数字,按Enter键继续...

图 6: open bin file