

# 操作系统原理

## 实验报告

实验题目： 实验 6：内存分配与回收

实验时间： 2022 年 12 月 22 日

学号姓名： 吕强 22129390

# 实验目的和要求

- 1.加深对内存分配原理的理解。
- 2.深入了解如何分配和回收内存。

# 实验内容

设计并实现一个简单的内存分配与回收程序。在 Visual C++ 6.0 集成开发环境下，使用 C 语言编写程序实现并进行测试。

# 实验原理与提示

本实验主要针对操作系统中内存管理相关理论进行实验,要求实验者编写一个程序,该程序管理一块虚拟内存,实现内存分配和回收功能。

- (1) 设计内存分配的数据结构;
- (2) 设计内存分配函数;
- (3) 设计内存回收函数。

# 实验程序

```
#include "malloc.h"
#include "stdio.h"
#include <stdio.h>
#include "stdlib.h"
#define n 10
#define m 10
#define minisize 100 //最小长度

struct
{
    float address;//首地址
```

```
    float length;  
    int flag;  
}used_table[n];
```

```
struct  
{  
    float address;  
    float length;  
    int flag;  
}free_table[m];
```

```
void allocate( char J, float xk)//内存分配  
{  
    int i, k;  
    float ad;  
    k = -1;  
    for (i = 0; i<m; i++)//遍历空闲分区  
        if (free_table[i].length >= xk&&free_table[i].flag == 1)  
            if (k == -1 || free_table[i].length<free_table[k].length)  
                k = i;  
    if (k == -1)  
    {  
        printf("无可空闲区\n");  
        return;  
    }  
  
    if (free_table[k].length - xk <= minisize)  
    {  
        free_table[k].flag = 0;  
        ad = free_table[k].address;  
        xk = free_table[k].length;  
    }  
    else  
    {  
        free_table[k].length = free_table[k].length - xk;  
        ad = free_table[k].address + free_table[k].length;  
    }  
  
    i = 0;  
    while (used_table[i].flag != 0 && i<n)  
        i++;  
    if (i >= n)  
    {  
        printf("无表目填写已分分区， 错误\n");  
    }  
}
```

```

        if (free_table[k].flag == 0)
            free_table[k].flag = 1;
        else
        {
            free_table[k].length = free_table[k].length + xk;
            return;
        }
    }
    else
        used_table[i].address = ad;
    used_table[i].length = xk;
    used_table[i].flag = J;
return;
}

void reclaim( char J)
{
    int i, k, j, s, t;
    float S, L;
    s = 0;
    while ((used_table[s].flag != J || used_table[s].flag == 0) && s < n)
        s++;
    if (s >= n)
    {
        printf("找不到该作业\n");
        return;
    }
    used_table[s].flag = 0;
    S = used_table[s].address;
    L = used_table[s].length;
    j = -1; k = -1; i = 0;
    while (i < m && (j == -1 || k == -1))
    {
        if (free_table[i].flag == 1)
        {
            if (free_table[i].address + free_table[i].length == S) k = i; if
            (free_table[i].address == S + L) j = i;
        }
        i++;
    }
    if (k != -1)
        if (j != -1)
        {
            free_table[k].length = free_table[j].length + free_table[k].length + L;

```

```

        free_table[j].flag = 0;
    }
    else
        free_table[k].length = free_table[k].length + L;
else
    if (j != -1)
    {
        free_table[j].address = S;
        free_table[j].length = free_table[j].length + L;
    }
    else
    {
        t = 0;
        while (free_table[t].flag == 1 && t < m)
            t++;
        if (t >= m)
        {
            printf("主存空闲表没有空间,回收空间失败\n");
            used_table[s].flag = J;
            return;
        }
        free_table[t].address = S;
        free_table[t].length = L;
        free_table[t].flag = 1;
    }
    return;
}

int main()
{
    int i, a;
    float xk;
    char J;
    free_table[0].address = 10240;
    free_table[0].length = 102400;
    free_table[0].flag = 1;
    for (i = 1; i < m; i++)
        free_table[i].flag = 0;
    for (i = 0; i < n; i++)
        used_table[i].flag = 0;
    while (1)
    {
        printf("选择功能项 (0-退出,1-分配主存,2-回收主存,3-显示主存) \n");
        printf("选择功项(0~3) :");
    }
}

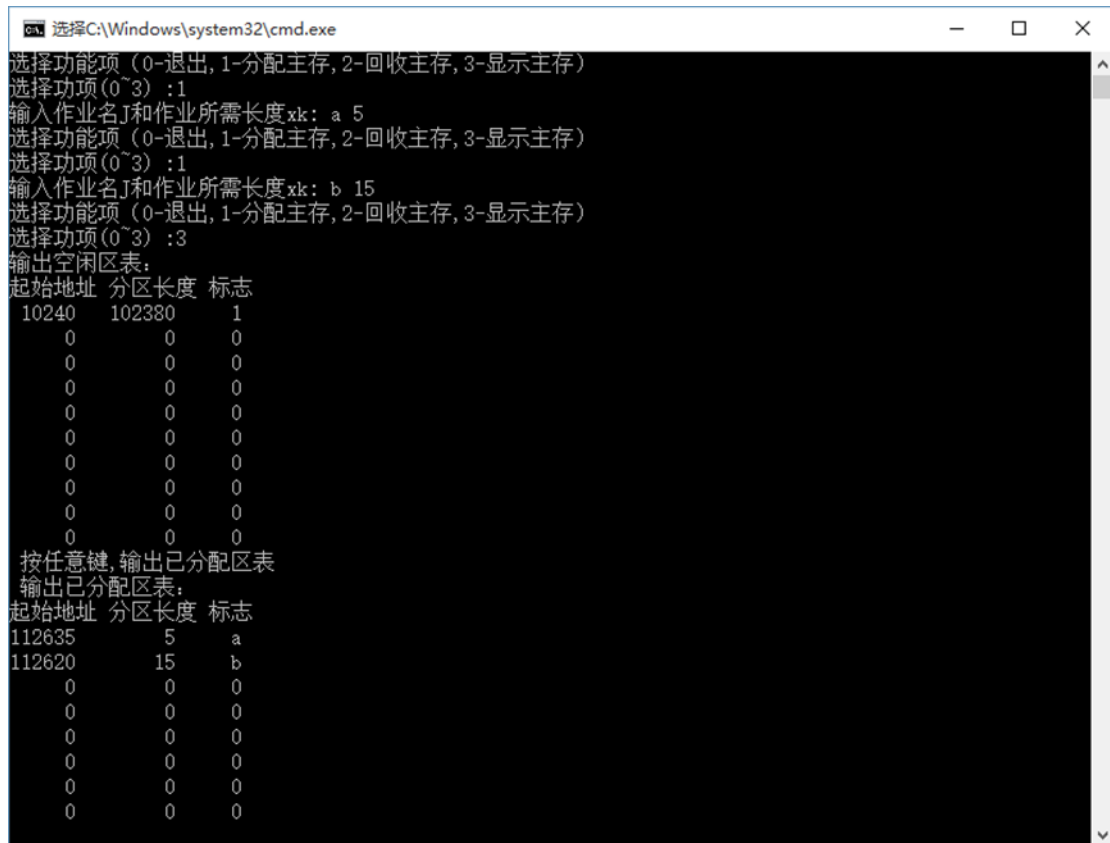
```

```

scanf("%d", &a);
switch (a)
{
case 0: exit(0);
case 1:
    printf("输入作业名 J 和作业所需长度 xk: ");
    scanf("%*c%c%f", &J, &xk);
    allocate(J, xk);
    break;
case 2:
    printf("输入要回收分区的作业名");
    scanf("%*c%c", &J);
    reclaim(J);
    break;
case 3:
    printf("输出空闲区表: \n 起始地址 分区长度 标志\n");
    for (i = 0; i < m; i++)
        printf("%6.0f%9.0f%6d\n", free_table[i].address, free_table[i].length,
free_table[i].flag);
    printf(" 按任意键,输出已分配区表\n");
    getchar();
    printf(" 输出已分配区表: \n 起始地址 分区长度 标志\n");
    for (i = 0; i < n; i++)
        if (used_table[i].flag != 0)
            printf("%6.0f%9.0f%6c\n", used_table[i].address,
used_table[i].length, used_table[i].flag);
        else
            printf("%6.0f%9.0f%6d\n", used_table[i].address,
used_table[i].length, used_table[i].flag);
    break;
default: printf("没有该选项\n");
}
}
return 0;
}

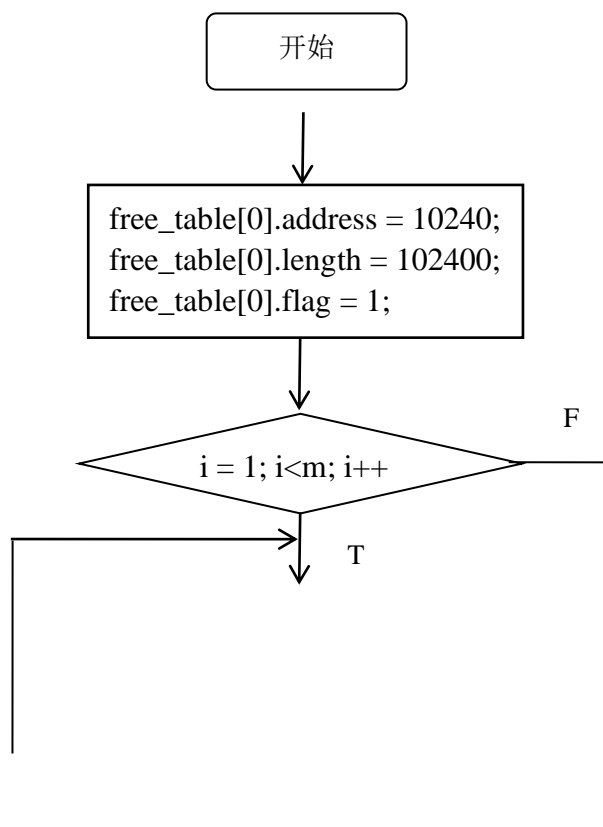
```

## 运行结果截图

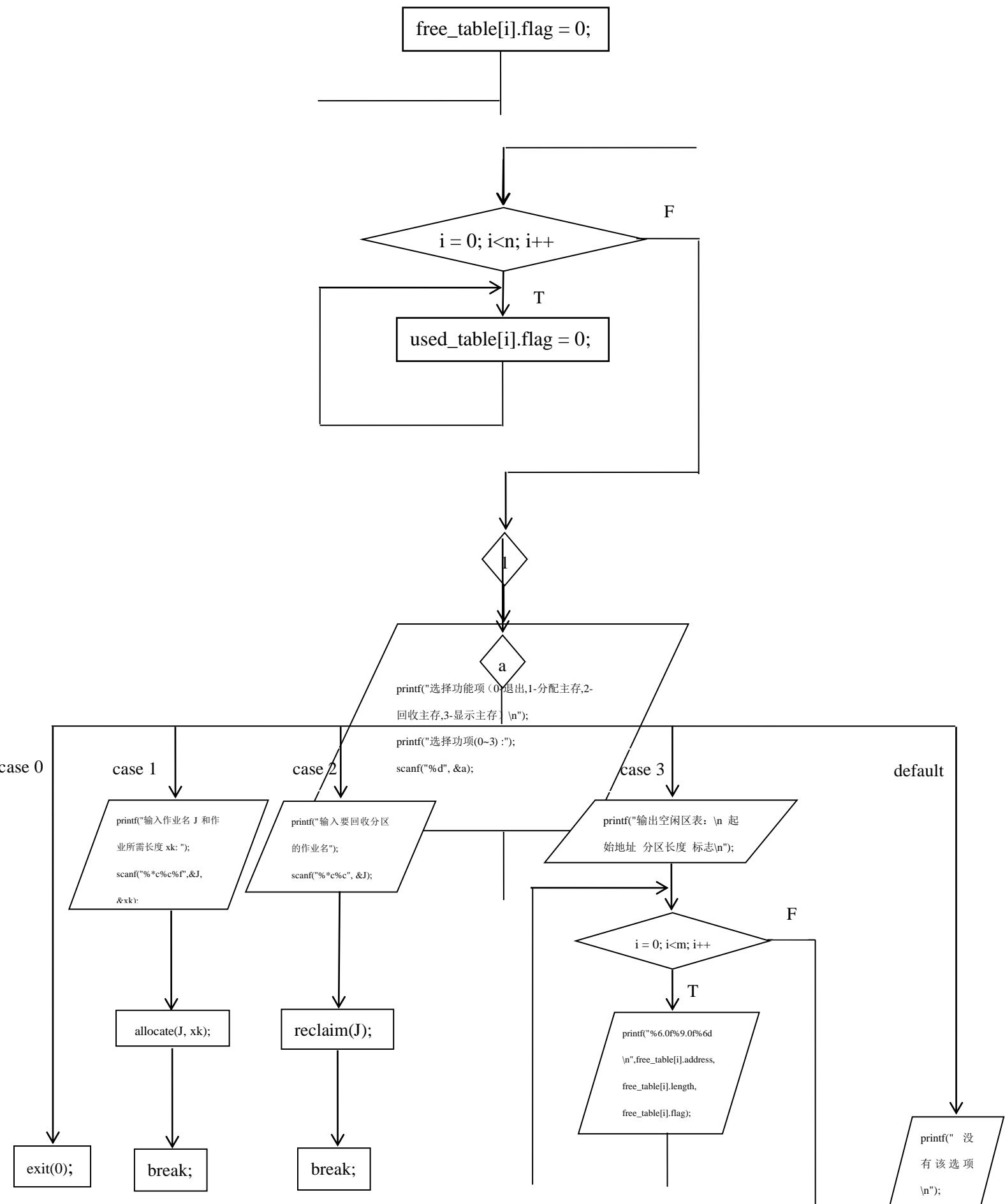


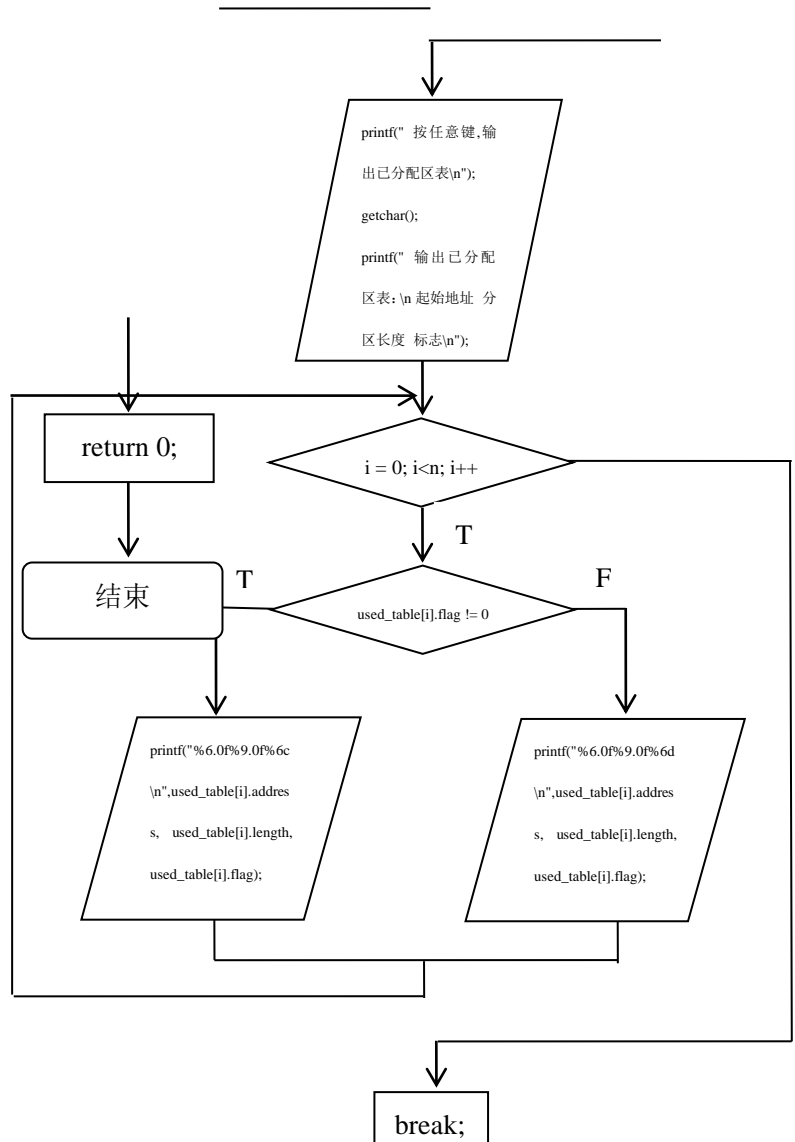
```
选择C:\Windows\system32\cmd.exe
选择功能项 (0-退出, 1-分配主存, 2-回收主存, 3-显示主存)
选择功项 (0~3) :1
输入作业名J和作业所需长度xk: a 5
选择功能项 (0-退出, 1-分配主存, 2-回收主存, 3-显示主存)
选择功项 (0~3) :1
输入作业名J和作业所需长度xk: b 15
选择功能项 (0-退出, 1-分配主存, 2-回收主存, 3-显示主存)
选择功项 (0~3) :3
输出空闲区表:
起始地址 分区长度 标志
10240 102380 1
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
按任意键, 输出已分配区表
输出已分配区表:
起始地址 分区长度 标志
112635 5 a
112620 15 b
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
0 0 0
```

## 主函数流程图









## 分析讨论

通过本次实验,了解到了内存的分配与回收的基本原理,在内存分配中,请求分区大小应该小于空闲分区大小.

在回收内存时回收区与空闲分区将组成一个新的空闲分区.

## 教师评语及成绩