**实验三 代码阅读并调试实验**

**任务：**

**1.阅读下面源代码，完善程序空白处内容。**

**2.阅读代码，写出源程序采用的调度算法、算法流程图和程序功能。**

**3.调试并运行代码，写出结果。**

**存储管理源程序如下：**

#include<stdio.h>

#include <dos.h>

#include<stdlib.h>

#include<conio.h>

//#include<iostream.h>

#define n 10 /\*假定系统允许的最大作业数为n，假定模拟实验中n值为10\*/

#define m 10 /\*假定系统允许的空闲区表最大为m，假定模拟实验中m值为10\*/

#define minisize 100 /\*空闲分区被分配时，如果分配后剩余的空间小于minisize，则将该空闲分区全部分配，若大于minisize，则切割分配\*/

struct

{

float address; /\*已分配分区起始地址\*/

float length; /\*已分配分区长度，单位为字节\*/

int flag; /\*已分配区表登记栏标志，用"0"表示空栏目\*/

} used\_table[n]; /\*已分配区表\*/

struct

{

float address; /\*空闲区起始地址\*/

float length; /\*空闲区长度，单位为字节\*/

int flag; /\*空闲区表登记栏标志，用"0"表示空栏目，用"1"表示未分配\*/

} free\_table[m]; /\*空闲区表\*/

void allocate(char J,float xk) /\*给J作业，采用最佳分配算法分配xk大小的空间\*/

{

int i,k;

float ad;

k=-1;

for(i=0; i<m; i++) /\*1 \*/

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)/\*未找到可用空闲区，返回\*/

{

printf("无可用空闲区\n");

return;

}

/\*找到可用空闲区，开始分配：若空闲区大小与要求分配的空间差小于minisize大小，则空闲区全部分配；若空闲区大小与要求分配的空间差大于minisize大小，则从空闲区划出一部分分配\*/

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;

}

/\*修改已分配区表\*/

i=0;

while(used\_table[i].flag!=0&&i<n) /\*2 \*/

i++;

if(i>=n) /\*无表目可填写已分配分区\*/

{

printf("无表目填写已分分区，错误\n");

/\*修正空闲区表\*/

if(free\_table[k].flag==0)

/\*前面找到的是整个空闲分区\*/

free\_table[k].flag=1;

else

{

/\*前面找到的是某个空闲分区的一部分\*/

free\_table[k].length=free\_table[k].length+xk;

return;

}

}

else

{

/\*修改已分配表\*/

used\_table[i].address=ad;

used\_table[i].length=xk;

used\_table[i].flag=J;

}

return;

}/\*主存分配函数结束\*/

void reclaim(char J)

/\*回收作业名为J的作业所占主存空间\*/

{

int i,k,j,s,t;

float S,L;

/\*寻找已分配表中对应登记项\*/

s=0;

while((used\_table[s].flag!=J||used\_table[s].flag==0)&&s<n)

s++;

if(s>=n)/\*在已分配表中找不到名字为J的作业\*/

{

printf("找不到该作业\n");

return;

}

/\*修改已分配表\*/

used\_table[s].flag=0;

/\*取得归还分区的起始地址S和长度L\*/

S=used\_table[s].address;

L=used\_table[s].length;

j=-1;

k=-1;

i=0;

/\*寻找回收分区的空闲上下邻，上邻表目k，下邻表目j\*/

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;/\*找到上邻\*/

if(free\_table[i].address==S+L)j=i;/\*找到下邻\*/

}

i++;

}

if(k!=-1)

if(j!=-1)

/\* 3 \*/

{

free\_table[k].length=free\_table[j].length+free\_table[k].length+L;

free\_table[j].flag=0;

}

else

/\*上邻空闲区，下邻非空闲区，与上邻合并\*/

free\_table[k].length=free\_table[k].length+L;

else if(j!=-1)

/\*上邻非空闲区，下邻为空闲区，与下邻合并\*/

{

free\_table[j].address=S;

free\_table[j].length=free\_table[j].length+L;

}

else

/\*上下邻均为非空闲区，回收区域直接填入\*/

{

/\*在空闲区表中寻找空栏目\*/

t=0;

while(free\_table[t].flag==1&&t<m)

t++;

if(t>=m)/\*空闲区表满,回收空间失败,将已分配表复原\*/

{

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;

}/\*主存回收函数结束\*/

int main( )

{

int i,a;

float xk;

char J;

/\*空闲分区表初始化：\*/

free\_table[0].address=10240; /\*起始地址假定为10240\*/

free\_table[0].length=10240; /\*长度假定为10240，即10k\*/

free\_table[0].flag=1; /\*初始空闲区为一个整体空闲区\*/

for(i=1; i<m; i++)

free\_table[i].flag=0; /\*其余空闲分区表项未被使用\*/

/\*已分配表初始化：\*/

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

used\_table[i].flag=0; /\*初始时均未分配\*/

while(1)

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\t\t选择功能项\n");

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("\n\n\t\t0-退出");

printf("\n\t\t1-分配主存");

printf("\n\t\t2-回收主存");

printf("\n\t\t3-显示主存");

printf("\n\n\t\t\t\t选择功能项(0~3) :\n");

scanf("%d",&a);

switch(a)

{

case 0:

exit(0); /\*a=0程序结束\*/

case 1: /\*a=1分配主存空间\*/

printf("输入作业名J和作业所需长度xk: \n");

scanf("%\*c%c%f",&J,&xk);

allocate(J,xk); /\*分配主存空间\*/

break;

case 2: /\*a=2回收主存空间\*/

printf("输入要回收分区的作业名");

scanf("%\*c%c",&J);

reclaim(J); /\*回收主存空间\*/

break;

case 3: /\*a=3显示主存情况\*/

/\*输出空闲区表和已分配表的内容\*/

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");

getch();

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");

}/\*case\*/

}/\*while\*/

return 1;

}/\*主函数结束\*/