**课题1参考：**

**（各位同学，以下代码仅供参考，并非完全正确。大家应该多做调查，按照自己的想法来设计算法流程以及编写代码，并给出测试结果。）**

Code：

#include <iostream>

using namespace std;

#define FREE 0

#define BUSY 1

#define MAX\_length 640

typedef struct freeArea//首先定义空闲区分表结构

{

int flag;

int size;

int ID;

int address;

}Elemtype;

typedef struct Free\_Node

{

Elemtype date;

struct Free\_Node \*front;

struct Free\_Node \*next;

}Free\_Node,\*FNodeList;

FNodeList block\_first;

FNodeList block\_last;

int alloc(int tag);//内存分配

int free(int ID);//内存回收

int best\_fit(int ID,int size);//最佳适应算法

void show();//查看分配

void init();//初始化

void Destroy(Free\_Node \*p);//销毁节点

void menu();

void init()//

{

block\_first=new Free\_Node;

block\_last = new Free\_Node;

block\_first->front=NULL;

block\_first->next=block\_last;

block\_last->front=block\_first;

block\_last->next=NULL;

block\_last->date.address=0;

block\_last->date.flag=FREE;

block\_last->date.ID=FREE;

block\_last->date.size=MAX\_length;

}

//实现内存分配

int alloc()

{

int ID,size1;

cout<<"请输入作业号：";

cin>>ID;

cout<<"请输入所需内存大小：";

cin>>size1;

if (ID<=0 || size1<=0)

{

cout<<"ERROR,请输入正确的ID和请求大小"<<endl;

return 0;

}

if (best\_fit(ID,size1))

{

cout<<"分配成功！"<<endl;

}

else cout<<"分配失败！"<<endl;

return 1;

}

int best\_fit(int ID,int size)//最佳适应算法

{

int surplus;//记录可用内存与需求内存的差值

FNodeList temp=(FNodeList)malloc(sizeof(Free\_Node));

Free\_Node \*p=block\_first->next;

temp->date.ID=ID;

temp->date.size=size;

temp->date.flag=BUSY;

Free\_Node \*q=NULL;//记录最佳位置

while(p)//遍历链表，找到第一个可用的空闲区间将他给q

{

if (p->date.flag==FREE&&p->date.size>=size)

{

q=p;

surplus=p->date.size-size;

break;

}

p=p->next;

}

while(p)//继续遍历，找到更加合适的位置

{

if (p->date.flag==FREE&&p->date.size==size)

{

p->date.flag=BUSY;

p->date.ID=ID;

return 1;

break;

}

if (p->date.flag==FREE&&p->date.size>size)

{

if (surplus>p->date.size-size)

{

surplus=p->date.size-size;

q=p;

}

}

p=p->next;

}

if (q==NULL)//如果没有找到位置

{

return 0;

}

else//找到了最佳位置

{

temp->next=q;

temp->front=q->front;

temp->date.address=q->date.address;

q->front->next=temp;

q->front=temp;

q->date.size=surplus;

q->date.address+=size;

return 1;

}

}

int free(int ID)//主存回收

{

Free\_Node \*p=block\_first->next;

while(p)

{

if (p->date.ID==ID)//找到要回收的ID区域

{

p->date.flag=FREE;

p->date.ID=FREE;

//判断P与前后区域关系

if (p->front->date.flag==FREE&&p->next->date.flag!=FREE)

{

p->front->date.size+=p->date.size;

p->front->next=p->next;

p->next->front=p->front;

}

if (p->front->date.flag!=FREE&&p->next->date.flag==FREE)

{

p->date.size+=p->next->date.size;

if(p->next->next)

{

p->next->next->front=p;

p->next = p->next->next;

}

else p->next=p->next->next;

}

if(p->front->date.flag==FREE&&p->next->date.flag==FREE)

{

p->front->date.size+=p->date.size+p->next->date.size;

if(p->next->next)

{

p->next->next->front=p->front;

p->front->next=p->next->next;

}

else p->front->next=p->next->next;

}

if(p->front->date.flag!=FREE&&p->next->date.flag!=FREE)

{//

}

break;

}

p=p->next;

}

cout<<"回收成功！"<<endl;

return 1;

}

void Destroy(Free\_Node \*p)

{

}

void show()

{

cout<<"------------------"<<endl;

cout<<"内存分配情况"<<endl;

cout<<"------------------"<<endl;

Free\_Node \*p=block\_first->next;

while(p)

{

cout<<"分区号：";

if (p->date.ID==FREE)

cout<<"FREE"<<endl;

else cout<<p->date.ID;

cout<<"起始地址："<<p->date.address<<endl;

cout<<"内存大小："<<p->date.size<<endl;

cout<<"分区状态：";

if (p->date.flag==FREE)

cout<<"空闲"<<endl;

else

cout<<"已分配"<<endl;

cout<<"------------------"<<endl;

p=p->next;

}

}

void menu()//菜单

{

int tag=0;

int ID;

init();

cout<<"分区模拟："<<endl;

while(tag!=4)

{

cout<<"请输入要进行的操作："<<endl;

cout<<"1-添加作业，2-内存回收，3-显示内存状况，4-退出"<<endl;

cin>>tag;

switch(tag)

{

case 1:

alloc();

break;

case 2:

cout<<"请输入需要回收的ID号："<<endl;

cin>>ID;

free(ID);

break;

case 3:

show();

break;

}

}

}

void main()

{

menu();

}

运行范例：

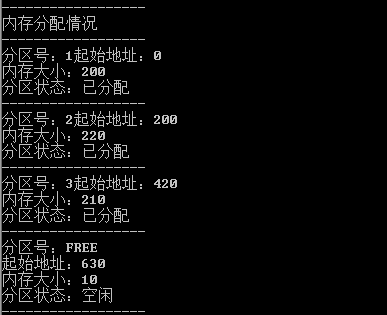
总内存空间设置为640KB

添加作业1 200KB、

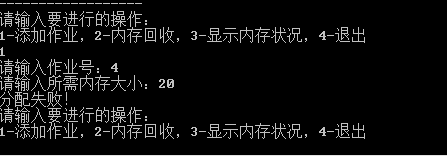
添加作业2 220KB

添加作业3 210KB

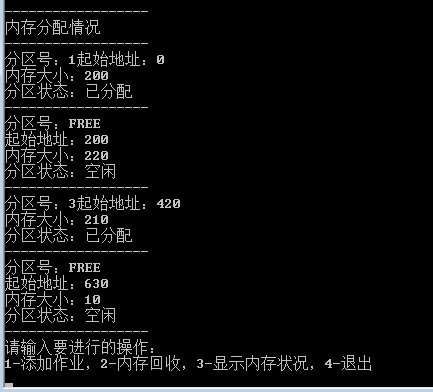
剩余空闲空间10KB



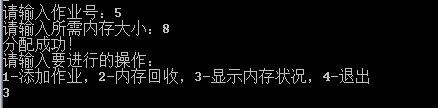
添加作业4 20KB，空闲内存空间不足，分配失败



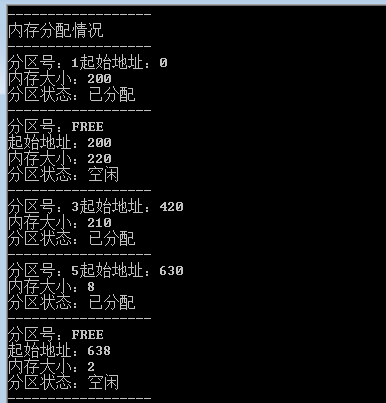
释放作业2，原分区2现在已经变为空闲空间



添加作业5 8KB



查看此时内存空间分配情况



可以看到，系统选择了较小的空闲空间分配给了作业5

最后退出系统

将空闲的内存区按其在储存空间中的起始地址递增的顺序排列，为作业分配储存空间时，从空闲区链的始端开始查找，选择第一个满足要求的空闲区，而不管它究竟有多大

优点:

1.在释放内存分区的时候，如果有相邻的空白区就进行合并，使其成为一个较大的空白区

2.此算法的实质是尽可能的利用储存器的低地址部分，在高地址部分则保留多的或较大的空白区，以后如果需要较大的空白区，就容易满足

缺点：

1. 在低地址部分很快集中了许多非常小的空白区，造成空间的浪费
2. 因而在空白区分配时，搜索次数增加，影响工作效率。