华 南 农 业 大 学 信 息（软 件） 学 院

《操作系统》综合性、设计性实验成绩单

开设时间：　　学年第　　学期

<table>
<tr><td>专业</td><td></td><td>班级</td><td></td><td>学号</td><td></td><td>姓名</td><td></td></tr>
<tr><td>实验题目</td><td colspan="7">（把你选的题目写在这里）</td></tr>
<tr><td>自我评价</td><td colspan="7">（即 实验体会和心得部分）</td></tr>
<tr><td>教师评语</td><td colspan="7">评价指标：
● 题目要求完成情况　优 □　良 □　中 □　差 □
● 对算法原理的理解程度　优 □　良 □　中 □　差 □
● 程序设计水平　优 □　良 □　中 □　差 □
● 实验报告结构清晰　优 □　良 □　中 □　差 □
● 流程图及内容表述清楚　优 □　良 □　中 □　差 □
● 实验总结和分析详尽　优 □　良 □　中 □　差 □</td></tr>
<tr><td>成绩</td><td colspan="3"></td><td>教师签名</td><td colspan="3"></td></tr>
</table>

一、需求分析

【**需求分析**：确定系统的目的、范围、定义和功能】

（1） 模拟操作系统在内存的分配以及回收所用内存的运行过程，分别采用首次适应法、最佳适应法和最差适应法来进行。

（2） 本程序的运行实际上不用输入测试数据，它能利用随机函数生成测试数据，对于用户来讲，可以直接观察运行结果，十分方便易用。

（3） 输出的形式是将内存分配形象成一个表，里面包括内存每个区的起址、长度、空闲状态、以及所分配的作业。对于用户来说，直观明瞭。

二、概要设计

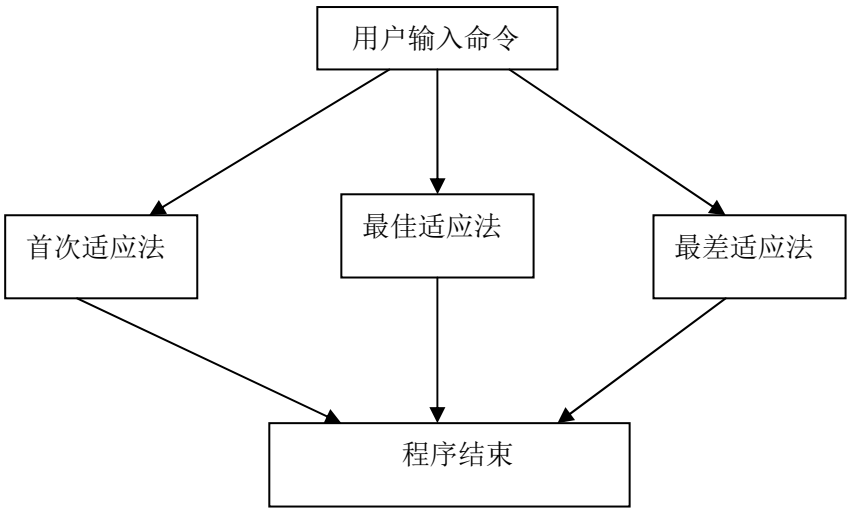
【**概要设计**：根据需求划分功能模块，描述各模块的流程；数据库或数据结构设计；用户界面设计等。这一步的设计是“概括的设计”，主要作用在于划分功能模块。】

1. 采用队列来分别模拟内存分配说明表和处于运行状态的进程队列。

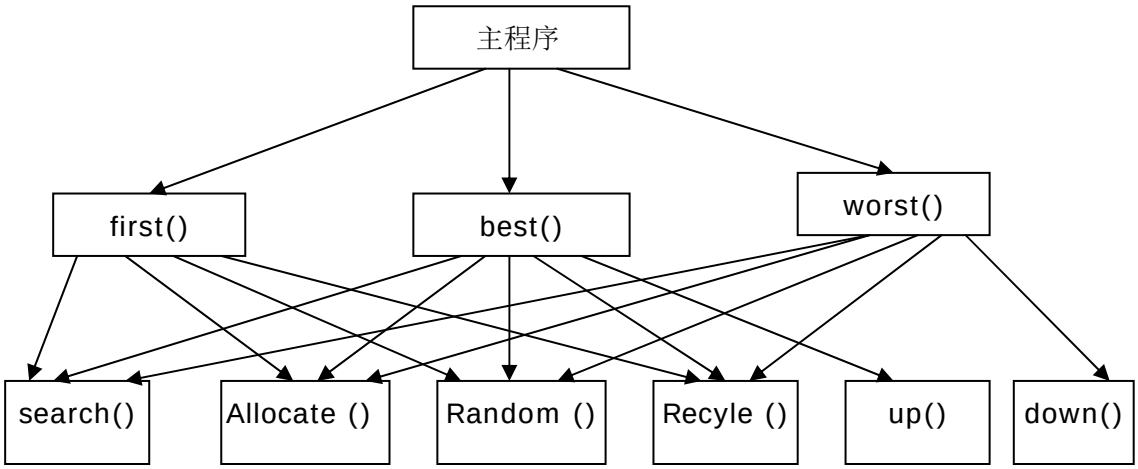
2. 本程序共 10 个模块，分别如下：

1) 初始化模块	random()
2) 内存回收模块	recyle()
3) 搜索空闲分区模块	search()
4) 对空闲分区进行升序排列模块	up()
5) 对空闲分区进行降序排列模块	down()
6) 随机分配作业模块	allocate()
7) 首次适应法模块	first()
8) 最佳适应法模块	best()
9) 最差适应法模块	worst()
10) 主程序模块	main()

3. 主程序的流程图



4. 各程序模块之间的层次关系



三、详细设计

【**详细设计**：根据概要设计，重点描述各模块的算法过程（用伪代码，而不是程序编码，一定要注意“详细设计不是编程序代码”）；数据结构设计的细化（详细说明实现各功能所需的数据、类、数据库中 SQL 语句等）；用户界面设计的细化（因为我们的实验简单，所以此部分可省略）。这一步是模块的详细规格说明。】

1． 程序中定义的抽象数据类型

```
struct table                    //模拟内存分配说明表
{
    int flag[10];   //记录分区的状态，若为空闲则置为－1,否则记录所分配作业的号数
    int base[10];       //记录分区的起址
    int length[10];     //记录分区的长度
    int free[10];     //记录空闲分区的序号
    int tot;          //记录目前分区总数
};

struct newprocess         //模拟处于运行状态的进程队列
{
    int v[10];      //进程大小
    int num[10];   //进程的 PID
};
```

2． int memory＝400；　　//内存的总空间，定义其大小为 400K

3． 初始化模块：random()

输入值：　无
返回值：　无
操作变量：　struct table tab;　　struct newprocess pro;
调用模块：　打印表单模块　　show();
详细设计：　（伪代码）

```
        用户启动程序
        申请内存空闲区域表单元；
```

随机设定作业大小
建立内存空闲区域表并初始化；
打印该表单；

4．内存回收模块 recyle()
输入值：　无
返回值：　无
操作变量：　struct table tab;　struct newprocess pro;
调用模块：　打印表单模块　show();
详细设计：
For(；次数不超过 3 次；)
根据先进先出原则对作业进行出队操作，回收；
for(;范围不超过总分区数;)
{
if（当作业号数匹配时）
{
回收内存；
if（其前一个分区也为空闲时）
{
进行向前合并操作；
合并后指针前移；
总分区数减 1；
}
if（其后一个分区也为空闲时）
{
进行向后合并操作；
总分区数减 1；
}
跳出循环；
}

}
打印内存分配列表；

5．搜索空闲分区模块 search()
输入值：　无
返回值：　1 或 0
操作变量：　struct table tab;
调用模块：　无
详细设计：
For(;区域不超过总分区数；)
{
if（内存分区为空）
{
记录空闲分区的序号数；
数组首元素用来记录空闲分区个数；
作好标记

}
}
If(flag 不为 0) 返回 1；
Else　返回 0；

6．对空闲分区进行升序排列模块 up()
输入值：　无
返回值：　无
操作变量：　struct table tab;
调用模块：　无
详细设计：　对空闲分区进行冒泡法排序；
　　记录在 tab.free[]中；

7．对空闲分区进行降序排列模块 down()
输入值：　无
返回值：　无
操作变量：　struct table tab;
调用模块：　无
详细设计：　对空闲分区进行冒泡法排序；
　　记录在 tab.free[]中；

8．随机分配作业模块 allocate()
输入值：　无
返回值：　无
操作变量：　struct table tab;　struct newprocess pro;
调用模块：　打印表单模块　show();
详细设计：

```
Do{
For(；不超过空闲分区总数；)
{
    if（空闲区大小大于等于作业大小时）
    {
        if（空闲区大小大于作业大小时）
            {
            其后的分区要后移；
            分割空闲区；
            总分区数加 1；
            }
            Else if（空闲区大小等于作业大小时）
            {
            作业直接进驻内存；
            }
        打印内存分配情况表；
        跳出循环；
      }
      分配标志记为 1；
 }
 }while(分配标志不为 1)；
```

9．首次适应法模块 first()
输入值：　无
返回值：　无
操作变量：　struct table tab;　struct newprocess pro;
调用模块：　打印表单模块 show();内存回收模块 recyle()；搜索空闲分区模块 search()；随机分配作业模块 allocate()；

详细设计：

```
Do{
For(；不超过 3；)
{
        回收内存；
        搜索空闲分区；
        随机分配作业；
        用户输入命令；
    }
    }while(用户要求继续执行)；
```

10. 最佳适应法模块 best()

输入值： 无

返回值： 无

操作变量： struct table tab; struct newprocess pro;

调用模块： 打印表单模块 show();内存回收模块 recyle()；搜索空闲分区模块 search()；升序排列内存空闲分区模块 up();随机分配作业模块 allocate()；

详细设计：

```
Do{
For(；不超过 3；)
{
        回收内存；
        搜索空闲分区；
        按大小升序排列内存空闲分区；
        随机分配作业；
        用户输入命令；
    }
    }while(用户要求继续执行)；
```

11. 最差适应法模块 worst()

输入值： 无

返回值： 无

操作变量： struct table tab; struct newprocess pro;

调用模块： 打印表单模块 show();内存回收模块 recyle()；搜索空闲分区模块 search()；降序排列内存空闲分区模块 down();随机分配作业模块 allocate()；

详细设计：

```
Do{
For(；不超过 3；)
{
        回收内存；
        搜索空闲分区；
        按大小降序排列内存空闲分区；
        随机分配作业；
        用户输入命令；
    }
    }while(用户要求继续执行)；
```

四、调试分析

（1）调试过程中遇到的问题是如何解决的以及对设计与实现的讨论和分析

每次产生的随机数都是一样的：产生错误的原因是放置随机函数句的位置错误。

对内存的回收有时会出错，例如有两个连续的空闲分区在内存表的末尾，这样在合

并时就会出错。解决方法是修改分区前移的功能代码。

对于用户命令的录入和识别问题，程序会出现异常反应，如菜单的重复出现。解决方法是不使用 C 的输入函数，而使用 C++下的输入函数，使问题迎刃而解。

（2）算法的时间复杂性和改进设想

本程序中内存回收模块、 随机分配作业模、对空闲分区进行升序排列模块和对空闲分区进行降序排列模块的时间复杂度都是 O(n^2)，初始化模块、搜索空闲分区模块的时间复杂度是 O(n)，其余的模块都是调用上述的功能模块，时间复杂度则根据所调用模块而定。

改进设想：

因为本程序所进行主要操作就是分区的前移和后移，若选择动态链表作为主要的数据结构而不是结构体数组的话，时间复杂度就可以有所降低。但是为了更形象地模拟内存分配的情况，而不是仅从时间复杂性角度出发，还是决定使用数组来做。虽然有点繁琐，但其实这对于我们理解内存分配还是有很大帮助的。

（3） 设计过程的经验和体会

内存分配的这个题目对我来说并不陌生，因为上课时候就曾听过老师很详细的讲解，加上课后自己对书本的回顾，对它有比较深的认识。开始本来打算由用户设定作业的次数和每个作业的大小，但老师强调最好是由计算机随机生成这些变量，所以就选择运用随机函数。对于作业结束时刻这个问题，我本来打算利用随机生成每个作业的运行时间，但是若是这样就可能产生某个作业长期占用内存的问题，虽然这对于计算机来说是常见的，但是对于这个模拟内存分配的实验来说，可能会造成演示效果不佳的后果。所以我最终还是选择作业先进先出，每个作业运行时间相同这个方案来实现，这样可以给用户一个直观清晰的演示效果，从而增加系统的友好度。

经过这次操作系统的实验，我对数组的操作熟练程度又有了一个很大的提升。虽然如果用指针代替数组可以令效率有所提高，但是 C 语言的特点就是灵活，用不同的数据结构都可以解决问题。也许用指针可以轻松解决问题，但我选择了数组，这也令我获益良多。

（4） 实现过程中出现的主要问题及解决方法

对程序各模块的定义开始并没有一个很清晰的概念，导致分类不清。我最初的程序设计中，分类并不够细致，没有将这个问题细化，所以曾经变成效率较低，也令程序变得十分繁琐冗余，后来我重新对这一部分作了分析，划分了较多的情况，程序变得明了清晰。

另外，为了让用户更易操作，程序对全部所需数据都是随机生成，用户只需选择继续或退出，就可观看整个操作过程，可以说这是一个自动的模拟内存分配的演示程序，更迎合大部分用户的要求。在实现过程中需注意问题还是随机数范围界定，经过细心的分析和计算，可以确保程序的运行效果清晰明显。

五、用户使用说明

本程序的功能是实现一个自动模拟内存分配情况的系统。

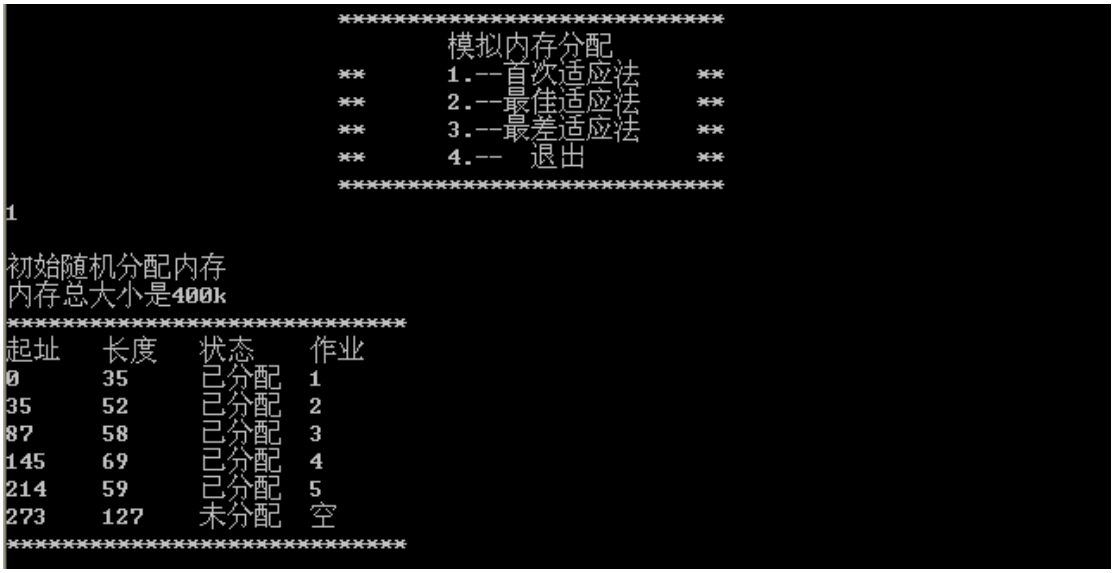
程序开始要求用户选用哪一种算法进行内存分配。用户需按数字键进行选择，并按回车确定。系统此时会随机生成作业进驻内存，之后进行内存回收，随机生成作业并按照所选算法进行内存分配。选择“退出”则结束程序。

在分配 3 个作业后，系统出作出提示询问用户是否继续运行内存分配程序，若用户希望继续，则键入’Y’或’y’，按回车确定；若想退出，则键入’N’或’n’，也需要按回车确定。选择继续，系统将继续分配作业进驻内存；选择退出，则返回主菜单。

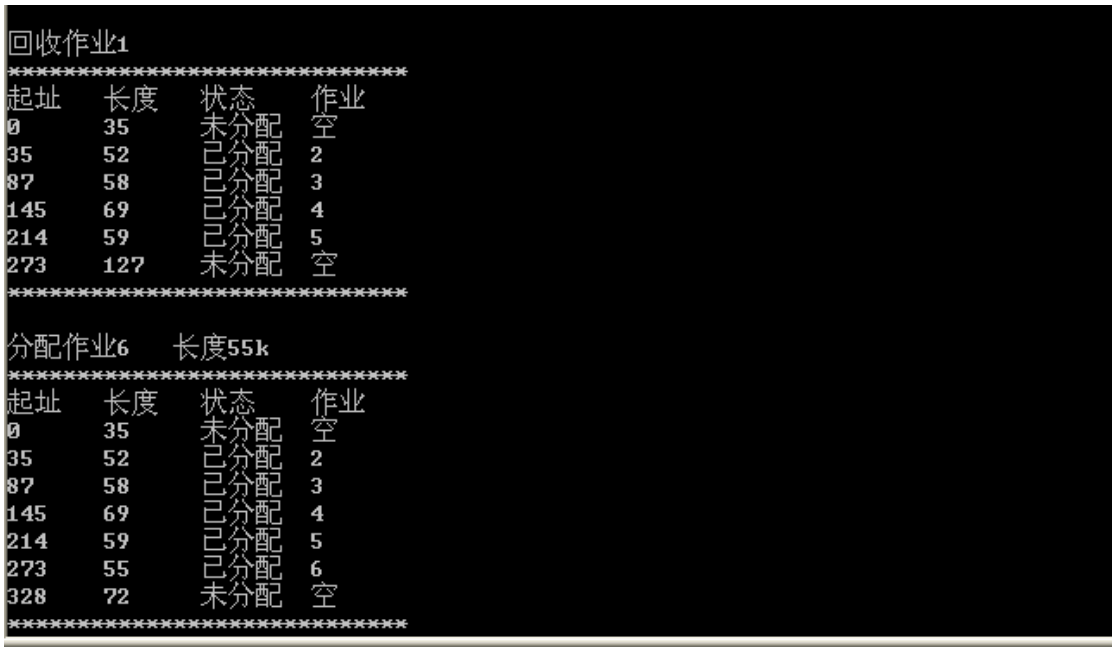
六、测试与运行结果

主菜单

初始化内存，开始随机分配 5 个作业进驻内存



按照先进先出原则结束作业，回收内存，并随机分配作业进驻内存



程序询问用户是否继续，用户按提示键入命令，回车确定