**实验二 作业调度-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，写出源程序采用什么调度算法、算法流程图和程序功能。**

**3.解释作业控制块构JCB的定义和作用。**

**4.为main()写出每行的注释。**

**5.调试并运行代码，写出结果。**

**作业调度源程序如下：**

#define \_CRT\_SECURE\_NO\_WARNINGS 1

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<conio.h>

#define getpch(type) (type\*)malloc(sizeof(type))

int n;

float T1=0,T2=0;

int times=0;

struct jcb      //作业控制块

{

    char name[10];  // **？添加注释**

    int reachtime;   //**？** **添加注释**

    int starttime;    //**？**  **添加注释**

    int needtime;       //**？添加注释**

    int finishtime;       //作业完成时间

    float cycletime;       //作业周转时间

    float cltime;           //？**添加注释**

    char state;            //作业状态

    struct jcb \*next;      //结构体指针

}\*ready=NULL,\*p,\*q;

typedef struct jcb JCB;

void inital()   //建立作业控制块队列,先将其排成先来先服务的模式队列

{

    int i;

    printf("\n输入作业数:");

    scanf("%d", &n);

    for (i = 0; i<n; i++)

    {

        p = getpch(JCB);

        printf("\n输入作业名:");

        scanf("%s", p->name);

        getchar();

        p->reachtime = i;

        printf("作业默认到达时间:%d", i);

        printf("\n输入作业要运行的时间:");

        scanf("%d", &p->needtime);

        p->state = 'W';

        p->next = NULL;

        if (ready == NULL)

        ready = q = p;

        else

        {

                q->next = p;

                q = p;

        }

    }

}

void disp(JCB\*q,int m) //显示作业运行后的周转时间及带权周转时间等

{

    printf("\n作业%s正在运行，估计其运行情况：\n",q->name);

    printf("开始运行时刻：%d\n",q->starttime);

    printf("完成时刻：%d\n",q->finishtime);

    printf("周转时间：%f\n",q->cycletime);

    printf("带权周转时间：%f\n",q->cltime);

    getchar();

}

void running(JCB \*p, int m)  //运行作业

{

    if (p == ready)     //先将要运行的作业从队列中分离出来

    {

    ready = p->next;

    p->next = NULL;

    }

    else

    {

        q = ready;

        while (q->next != p)  q = q->next;

        q->next = p->next;

    }

    p->starttime = times;    //计算作业运行后的完成时间,周转时间等等

    p->state = 'R';

    p->finishtime = p->starttime + p->needtime;

    p->cycletime = (float)(p->finishtime - p->reachtime);

    p->cltime = (float)(p->cycletime / p->needtime);

    T1 += p->cycletime;

    T2 += p->cltime;

    disp(p, m);        //调用disp()函数,显示作业运行情况

    times += p->needtime;

    p->state = 'F';

    printf("\n%s has been finished!\n", p->name);

    free(p);          //释放运行后的作业

    getchar();

}

void final() //最后打印作业的平均周转时间,平均带权周转时间

{

    float s,t;

    t=T1/n;

    s=T2/n;

    getchar();

    printf("\n\n作业已经全部完成!\n");

    printf("\n%d个作业的平均周转时间是：%f",n,t);

    printf("\n%d个作业的平均带权周转时间是%f：\n\n\n",n,s);

}

void sjf(int m)      // 最短作业优先算法

{

    JCB \*min;

    int i,iden;

    system("cls");

    inital();

    for(i=0;i<n;i++)

    {

        p=min=ready;

        iden=1;

        do{

                if(p->state=='W'&&p->reachtime<=times)

                if(iden)

                {

                    min=p;

                    iden=0;

                }

                else if(p->needtime<min->needtime) min=p;

                 ; // **1 根据上下文，空格处填写一条指令**

            } while (p != NULL);

        if(iden)

        {

            i--;

            times++;

            if (times>100)

            {

             printf("\nruntime is too long … error");

             getchar();

            }

        }

        else

        {

            running(min, m);

        }

    }

    final();

}

void fcfs(int m)     //先来先服务算法

{

    int i, iden;

    system("cls");

    inital();

    for (i = 0; i<n; i++)

    {

        p = ready;

        iden = 1;

        do {

            if (p->state == 'W'&&p->reachtime <= times)  iden = 0;

            if (iden)p = p->next;

        } while (p != NULL&&iden);

       if (iden)

        {

            i--;

            printf("\n没有满足要求的进程,需等待");

             ; // **2根据上下文，填入一条语句**

            if (times>100)

            {

               printf("\n时间过长");

               getchar();

            }

        }

        else

        {

            running(p, m);

        }

    }

    final();

}

void menu()

{

    int m;

    system("cls");                   <br>    printf("\n\n\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\t\t\n");

    printf("\t\t\t\t作业调度演示\n");

    printf("\t\t\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\t\t\n");

    printf("\n\n\n\t\t\t1.先来先服务算法.");

    printf("\n\t\t\t2.最短作业优先算法.");

    printf("\n\t\t\t0.退出程序.");

    printf("\n\n\t\t\t\t选择所要操作:");

    scanf("%d", &m);

    switch (m)

    {

    case 1:

        fcfs(m);

        getchar();

        system("cls");

        break;

    case 2:

        sjf(m);

        getchar();

        system("cls");

        break;

        case 0:

        system("cls");

        break;

        default:

        printf("选择错误,重新选择.");

        getchar();

        system("cls");

        menu();

    }

}

int main()

{

    menu();

    system("pause");

    return 0;

}