МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

дисциплина «Операционные системы»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

«**Программирование планировщиков процессов**»

Выполнил студент гр. ИТИ-11:

Батюшков А.С.

#### Принял преподаватель:

#### Карась О.В.

Гомель 2022

**Цель:**Разработать программу, осуществляющую моделирование режима работы с разделение времени.

1. Имя
2. Продолжительность

**Задание на лабораторную работу.**

В соответствии с вариантом (выдается преподавателем), разработать программу на языке С# (или другом), осуществляющую моделирование работы планировщика процессов.

Моделирование режима разделения времени выполняется в соответствии с заданием (таблица 2.1). На экран выводится следующая информация:

1. Номер текущего кванта времени процессора;

2. Таблица процессов с указанием имени процессов, продолжительности, приоритета (в зависимости от задания), оставшегося времени выполнения, время появления;

3. Таблица планирования процессов с отображением текущего состояния процессов.

После запуска, программа должна диалоговом режиме, ввести информацию процессах – имя, длительность, приоритет, время появления. Для алгоритмов RR число квантов времени. Выполнение должно производиться в пошаговом режиме (по нажатию на кнопку).

Алгоритм FCFS. Не вытесняющий

#include<stdio.h>

#include<math.h>

#include<string.h>

#include<stdlib.h>

void getproces(int\*col,char\*Name,int\*Ti,int\*Pr,int\*Po,int\*Oc)

{

for(int i=0;i<\*col;i++)

{

printf("Enter process name N-%d: ",i+1);

scanf("%s",&(Name+i\*1000)[0]);

printf("Enter process duration: ");

scanf("%d",&Ti[i]);

printf("Enter process priority: ");

scanf("%d",&Pr[i]);

printf("Enter process entry time: ");

scanf("%d",&Po[i]);

Oc[i]=0;

}

}

int Twork(int\*col,int\*Ti)

{

int i,s=0;

for(i=0;i<\*col;i++)

{

s += \*(Ti+i);

}

return(s);

}

void Move(int\*och,int\*q,int\*col,int\*Save,int\*Ti,int\*Pr,int\*Po,int\*Oc)

{

int i,j,tmp,aim=0,ind,dec=0;

for(i=0;i<\*col;i++)

{

if(\*(Po+i)==\*q && \*(Oc+i)==0)

{

\*(Oc+i)=(\*och)++;

}

}

for(i=0;i<\*col;i++)

{

for(j=i+1;j<\*col;j++)

{

if(\*(Oc+i)!=-1 && \*(Oc+j)!=-1 && \*(Oc+i)!=0 && \*(Oc+j)!=0 && i!=j && \*(Pr+j)<\*(Pr+i) && \*(Oc+j)>\*(Oc+i) && \*(Oc+i)!=1 && \*(Oc+j)!=1 )

{

tmp=\*(Oc+i);

\*(Oc+i)=\*(Oc+j);

\*(Oc+j)=tmp;

}

}

}

if(\*och>1)

{

while(aim<=\*och)

{

if(aim==0)

{

aim++;

}

for(i=0;i<\*col;i++)

{

if(aim==\*(Oc+i))

{

ind=i;

i=\*col;

}

}

if(aim==1)

{

\*(Save+\*q\*100+ind)=2;

(\*(Ti+ind))--;

if(\*(Ti+ind)==0)

{

dec=1;

}

}

if(aim!=1 && \*(Oc+ind)==aim)

{

\*(Save+\*q\*100+ind)=1;

}

aim++;

}

for(i=0;i<\*col;i++)

{

if(\*(Oc+i)<1)

{

\*(Save+\*q\*100+i)=0;

}

}

if(dec==1)

{

(\*och)--;

for(i=0;i<\*col;i++)

{

if(\*(Oc+i)!=0)

{

(\*(Oc+i))--;

if(\*(Oc+i)==0)

{

(\*(Oc+i))--;

}

}

}

}

}

(\*q)++;

}

void AllTabl(int\*q,int\*col,int\*Save,char\*Name)

{

int i,j;

printf("\xDA\xC4\xC4\xC4\xC4\xC4\xC4\xC2");

for(i=1;i<\*col;i++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC2");

}

printf("\xC4\xC4\xC4\xC4\xC4\xBF\n");

printf("\xB3 Name \xB3");

for(i=0;i<\*col;i++)

{

printf("%3s \xB3",Name+i\*1000);

}

printf("\n\xC3\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

for(i=1;i<\*col;i++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC5");

}

printf("\xC4\xC4\xC4\xC4\xC4\xB4\n");

for(i=0;i<\*q;i++)

{

printf("\xB3 %2d \xB3",i);

for(j=0;j<\*col;j++)

{

if(\*(Save+i\*100+j)==0)

{

printf(" \xB3");

}

if(\*(Save+i\*100+j)==1)

{

printf(" R \xB3");

}

if(\*(Save+i\*100+j)==2)

{

printf(" E \xB3");

}

}

printf("\n");

if(i!=\*q-1)

{

printf("\xC3\xC4\xC4\xC4\xC4\xC4\xC4\xC5");

for(j=1;j<\*col;j++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC5");

}

printf("\xC4\xC4\xC4\xC4\xC4\xB4\n");

}

}

printf("\xC0\xC4\xC4\xC4\xC4\xC4\xC4\xC1");

for(i=1;i<\*col;i++)

{

printf("\xC4\xC4\xC4\xC4\xC4\xC1");

}

printf("\xC4\xC4\xC4\xC4\xC4\xD9\n");

}

int main()

{

int q=0,j,och=1,col,TimeL[1000],Prior[1000],Poivl[1000],Ochered[1000],work,Save[100][100];

char Name[1000][1000];

printf("Enter the number of processes: ");

scanf("%d",&col);

getproces(&col,&Name[0][0],&TimeL[0],&Prior[0],&Poivl[0],&Ochered[0]);

work=Twork(&col,&TimeL[0]);

while(work!=0)

{

Move(&och,&q,&col,&Save[0][0],&TimeL[0],&Prior[0],&Poivl[0],&Ochered[0]);

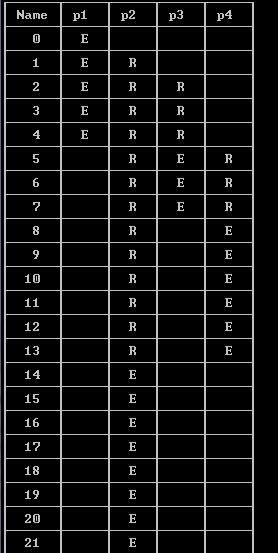
work=Twork(&col,&TimeL[0]);

}

AllTabl(&q,&col,&Save[0][0],&Name[0][0]);

return(0);

}



**Вывод:** Разработана программа, осуществляющая моделирование режима работы с разделение времени с использованием заданного алгоритма.