先来先服务：

#include <iostream>

#include <cstdlib>

#include <numeric>

using namespace std;

#define MAX 10

char process[MAX]=""; //进程标识

int arrivetime[MAX];//达到时间

int servicetime[MAX];//服务时间

int finishtime[MAX]; //完成时间

int turnovertime[MAX];//周转时间

double avgturnovertime; //平均周转时间

double powertime[MAX]; //带权周转时间

double avgpowertime; //平均带权周转时间

int init();

void FCFS();

void output();

void showsingle(int\* arr,int len);

//初始化，并返回进程数

int init()

{

cout << "输入进程队列标识(用单个字母表示一个进程,字母间用tab间隔)" << endl;

int i=0;

while(i<MAX)

{

cin.get(process[i]);

if(process[i]==' ' || process[i]=='\t')

{

continue;

}

if(process[i]=='q' || process[i]=='\n')

{

process[i]='\0';

break;

}

i++;

}

int len=strlen(process);

cout << "依次输入进程到达时间(时间之间用tab间隔)" << endl;

for(int ix=0; ix<len; ix++)

{

cin >> arrivetime[ix];

}

cout << "依次输入服务时间(时间之间用tab间隔)" <<endl;

for(ix=0; ix<len; ix++)

{

cin >> servicetime[ix];

}

return len;

}

void FCFS(int len)

{

//完成时间的计算

for(int ix=0; ix<len; ix++)

{

finishtime[ix]=accumulate(servicetime,servicetime+ix+1,0);

}

//周转时间计算

for(ix=0; ix<len; ix++)

{

turnovertime[ix]=finishtime[ix]-arrivetime[ix];

}

avgturnovertime=accumulate(turnovertime,turnovertime+len,0)\*1.0/len;

//带权周转时间计算

for(ix=0; ix<len; ix++)

{

powertime[ix]=turnovertime[ix]\*1.0/servicetime[ix];

}

//平均带权周转时间

double tmptotal=0.0;

for(ix=0; ix<len; ix++)

{

tmptotal+=powertime[ix];

}

avgpowertime=tmptotal/len;

}

void output()

{

cout <<endl<<endl;

cout <<"+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"<<endl;

int len=strlen(process);

//显示进程序列

for(int ix=0; ix<len; ix++)

{

cout <<process[ix] << "\t";

}

cout << endl;

//显示到达时间序列

showsingle(arrivetime,len);

//显示服务时间序列

showsingle(servicetime,len);

cout <<endl<<endl;

//显示完成时间序列

showsingle(finishtime,len);

//显示周转时间序列

showsingle(turnovertime,len);

cout << "平均周转时间 :" << avgturnovertime << endl;

//显示带权周转时间序列

for(ix=0; ix<len; ix++)

{

cout << powertime[ix] << "\t";

}

cout <<endl;

cout << "平均带权周转时间:" << avgpowertime << endl;

cout <<"+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++"<<endl;

}

//对int类型的数组进行格式化输出

void showsingle(int\* arr,int len)

{

for(int ix=0; ix<len; ix++)

{

cout << arr[ix] << "\t";

}

cout <<endl;

}

int main()

{

cout << "\t\t||本程序是先来先服务算法||" << endl;

int len = init();

FCFS(len);

output();

system("PAUSE");

return 0;

}

优先调度算法：

#include "stdio.h"

struct sjf{

char name[10];

float arrivetime;

float servicetime;

float starttime;

float finishtime;

float zztime;

float dqzztime;

};

sjf a[100];

void input(sjf \*p,int N)

{ int i;

printf("intput the process's name & arrivetime & servicetime:\nfor exmple: a 0 100\n");

for(i=0;i<=N-1;i++)

{

printf("input the %dth process's information:\n",i+1);

scanf("%s%f%f",&p[i].name,&p[i].arrivetime,&p[i].servicetime);

}

}

void Print(sjf \*p,float arrivetime,float servicetime,float starttime,float finishtime,float zztime,float dqzztime,int N)

{int k;

printf("run order:");

printf("%s",p[0].name);

for(k=1;k<N;k++)

{printf("-->%s",p[k].name);

}

printf("\nthe process's information:\n");

printf("\nname\tarrive\tservice\tstart\tfinish\tzz\tdqzz\n");

for(k=0;k<=N-1;k++)

{ printf("%s\t%-.2f\t%-.2f\t%-.2f\t%-.2f\t%-.2f\t%-.2f\t\n",p[k].name,p[k].arrivetime,p[k].servicetime,p[k].starttime,p[k].finishtime,p[k].zztime,p[k].dqzztime);

}

}

//pai xu

void sort(sjf \*p,int N)

{

for(int i=0;i<=N-1;i++)

for(int j=0;j<=i;j++)

if(p[i].arrivetime<p[j].arrivetime)

{

sjf temp;

temp=p[i];

p[i]=p[j];

p[j]=temp;

}

}

//yun xing jieduan

void deal(sjf \*p, float arrivetime,float servicetime,float starttime,float finishtime,float &zztime,float &dqzztime,int N)

{ int k;

for(k=0;k<=N-1;k++)

{

if(k==0)

{

p[k].starttime=p[k].arrivetime;

p[k].finishtime=p[k].arrivetime+p[k].servicetime;}

else

{

p[k].starttime=p[k-1].finishtime;

p[k].finishtime=p[k-1].finishtime+p[k].servicetime;}

}

for(k=0;k<=N-1;k++)

{

p[k].zztime=p[k].finishtime-p[k].arrivetime;

p[k].dqzztime=p[k].zztime/p[k].servicetime;

}

}

void sjff(sjf \*p,int N)

{float arrivetime=0,servicetime=0,starttime=0,finishtime=0,zztime=0,dqzztime=0;

sort(p,N);

for(int m=0;m<N-1;m++)

{if(m==0)

p[m].finishtime=p[m].arrivetime+p[m].servicetime;

else

p[m].finishtime=p[m-1].finishtime+p[m].servicetime;

int i=0;

for(int n=m+1;n<=N-1;n++)

{if(p[n].arrivetime<=p[m].finishtime)

i++;

}

float min=p[m+1].servicetime;

int next=m+1;//m+1=n

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

{

if(p[k+1].servicetime<min)

{min=p[k+1].servicetime;

next=k+1;}

}

sjf temp;

temp=p[m+1];

p[m+1]=p[next];

p[next]=temp;

}

deal(p,arrivetime,servicetime,starttime,finishtime,zztime,dqzztime,N);

Print(p,arrivetime,servicetime,starttime,finishtime,zztime,dqzztime,N);

}

void main()

{ int N;

printf("------短作业优先调度算法------\n");

printf("input the process's number:\n");

scanf("%d",&N);

input(a,N);

sjf \*b=a;

sjf \*c=a;

sjff(b,N);

}