```cpp
#include<iostream>
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<fstream>



using namespace std;
#define TAKEIN "进入"
#define WAIT "等待"
#define RUN "运行"
#define FINISH "结束"


typedef struct pcb
{

   char Name[20];
   int runTime;
   int endTime;
   int startTime;
   int turnOverTime;//周转时间
   int userweightTurnOverTime; //加权周转时间
   int arriverTime;
   char provessStatus[10];//进程状态


} pcb;

pcb pcbs[5];
int currentTime=0;  //时间
int processIndex=0; //进程编号


class mainPcb{
private:
    pcb pcbs1,pcbs2,pcbs3,pcbs4,pcbs5;
public:
    void initialPcb(); //初始化
    void printfPcbsInfo(); //打印所有进程的所有信息
    void sortPcbs(); //按到达时间升序排列
    int selNectProcess(); //下一个进程的选择，条件：等待状态和运行时间最短
    int isHasProcessArrive(); //检查在某个时间点有没有进程没有到达
    void runProcess(int pindex);  //运行
    void startProcess(); //开始
};
void mainPcb::initialPcb(){
    freopen("input.txt","r",stdin);
    //读出input.txt文件中的作业信息（包括进程名称、到达时间、运行时间）
    cout<<"进程名\t"<<"到达时间\t"<<"运行时间\n";

    for(int index=0; index<5; index++) //遍历所有进程并赋初值
    {
        cin>>pcbs[index].Name;
        cin>>pcbs[index].arriverTime;
        cin>>pcbs[index].runTime;
        pcbs[index].startTime=0;
        pcbs[index].endTime=0;
        pcbs[index].turnOverTime=0;
        pcbs[index].userweightTurnOverTime=0;
        strcpy(pcbs[index].provessStatus,TAKEIN);

        cout<<pcbs[index].Name<<"\t"<<pcbs[index].arriverTime<<"\t\t"<<pcbs[index].runTime<<"\n";

    }
}
void mainPcb::printfPcbsInfo()
{
    cout<<"当前时间为："<<currentTime<<"\n\n";

```

```
    cout<<"进程名"<<"\t"<<"到达时间"<<"\t"<<"运行时间"<<"\t"<<"开始时间"<<"\t"<<"结束
    时间"<<"\t"<<"周转时间"<<"\t"<<"带权周转时间"<<"\t"<<"状态"<<"\n";
71      for(int index=0; index<5;index++)
72      {
73          cout<<pcbs[index].Name<<"\t"<<pcbs[index].arriverTime<<"\t\t"
74          <<pcbs[index].runTime<<"\t\t"<<pcbs[index].startTime
75          <<"\t\t"<<pcbs[index].endTime<<"\t\t"<<pcbs[index].turnOverTime
76
            <<"\t\t"<<pcbs[index].userweightTurnOverTime<<"\t\t"<<pcbs[index].provessStatu
            s<<"\n";
77      }
78  }
79
80  void mainPcb::sortPcbs()  //排序
81  {
82    int minIndex=0,minValue=0;
83      for(int i=0; i<5; i++)
84      {
85          minIndex=i;
86          minValue=pcbs[i].arriverTime;
87          for(int j=i; j<5; j++)
88          {
89              if(pcbs[j].arriverTime<minValue)
90              {
91                  minValue=pcbs[j].arriverTime;
92                  minIndex=j;
93              }
94          }
95          pcb temp=pcbs[minIndex];
96          pcbs[minIndex]=pcbs[i];
97          pcbs[i]=temp;
98      }
99
100
101 }
102
103 int mainPcb::selNectProcess()
104 {
105    int result=-1;
106     int minTime=100;
107     for(int index=0; index<5; index++)
108     {
109         if(strcmp(pcbs[index].provessStatus,WAIT)==0)  //等待
110         {
111             if(pcbs[index].runTime<minTime)
112             {
113                 minTime=pcbs[index].runTime;
114                 result=index;
115             }
116         }
117     }
118     return result;
119 }
120 int mainPcb::isHasProcessArrive()
121 {
122     int result=-1;
123     for(int index=0; index<5; index++)
124     {
125         if(pcbs[index].arriverTime==currentTime)
126         {
127             result=index;
128             strcpy(pcbs[index].provessStatus,WAIT);
129         }
130     }
131     return result;
132
133
134 }
135 void mainPcb::runProcess(int pindex)
136 {
137
138   int runTime=pcbs[pindex].runTime;
139
```

```
140         pcbs[pindex].startTime=currentTime;
141         pcbs[pindex].endTime=pcbs[pindex].startTime+pcbs[pindex].runTime;
142         strcpy(pcbs[pindex].provessStatus,RUN);
143         printfPcbsInfo();
144         for(int k=1; k<=runTime; k++)
145         {
146             currentTime++;  //时间增加一个单位
147             isHasProcessArrive();
148             if(k==runTime)  //进程结束的条件
149             {
150                 strcpy(pcbs[pindex].provessStatus,FINISH);
151                 pcbs[pindex].turnOverTime=pcbs[pindex].endTime-pcbs[pindex].arriverTime;
152
                    pcbs[pindex].userweightTurnOverTime=pcbs[pindex].turnOverTime*1.0/pcbs[pin
                    dex].runTime;
153             }
154             printfPcbsInfo();  //打印该进程的信息
155         }
156         processIndex++;  //准备运行下一个进程
157         currentTime--;
158
159
160     }
161
162     void mainPcb::startProcess()
163     {
164       int firstArriveTime=pcbs[0].arriverTime;
165         int nextIndex=0;
166         printfPcbsInfo();
167         while(1)
168         {
169             currentTime++;
170             isHasProcessArrive();
171             if(currentTime<firstArriveTime)
172             {
173                 printfPcbsInfo();
174             }
175             else if(currentTime==firstArriveTime)
176             {
177                 runProcess(0);
178             }
179             else
180             {
181                 nextIndex=selNectProcess();
182                 if(nextIndex!=-1)
183                 {
184                     runProcess(nextIndex);
185                 }
186                 if(processIndex==5)
187                     break;
188             }
189         }
190
191     }
192
193     int main(){
194
                cout<<"=====================这是短作业优先调度算法====================="<<
                "\n\n";
195             mainPcb textpcb;
196             textpcb.initialPcb();
197
                cout<<"=================================================================="<
                <"\n";
198             textpcb.sortPcbs();
199             textpcb.startProcess();  //开始进程调度
200             return 0;
201     }
202
```