# RR轮转算法实验报告

实验目的：通过本实验加深理解有关进程队列的概念，体会时间片轮转调度算法的具体实施办法。

## 程序源代码:

头文件:

#pragma once

typedef struct PCB {

char name; *//进程名*

int ArrivalTime; *//到达时间*

int ServiceTime; *//服务时间*

int StartTime; *//开始时间*

int FinishTime; *//完成时间*

int ResidualTime; *//剩余服务时间*

double WholeTime; *//周转时间*

double weightWholeTime; *//带权周转时间*

int Priority; *//优先级*

}PCB;

typedef PCB DataType;

typedef struct Node {

DataType data;

struct Node \*next;

}Node;

typedef struct Queue {

Node \*front;

Node \*rear;

size\_t size; *//队列中有几个元素*

}Queue;

typedef struct linkList {

Node \*head;

Node \*tail;

size\_t size;

}linkList;

void QueueInit(Queue \*pqueue) { *//初始化*

assert(pqueue != NULL);

pqueue->front = NULL;

pqueue->rear = NULL;

pqueue->size = 0;

}

void QueueDestroy(Queue \*pqueue)

{

Node \*cur = NULL;

Node \*del = NULL;

assert(pqueue != NULL);

if (pqueue->front == NULL) {

return;

}

cur = pqueue->front;

while (cur != NULL) {

del = cur;

cur = cur->next;

free(del);

del = NULL;

}

}

Node \*CreatNode(DataType data)

{

Node \*ret = (Node \*)malloc(sizeof(Node));

if (ret == NULL) {

exit(0);

}

ret->data = data;

ret->next = NULL;

return ret;

}

void QueuePush(Queue \*pqueue, DataType data)

{

assert(pqueue != NULL);

Node \*temp = CreatNode(data);

if (pqueue->front == NULL) {

pqueue->front = temp;

pqueue->rear = temp;

*//pqueue->rear = pqueue->front = temp;*

++pqueue->size;

}

else {

pqueue->rear->next = temp;

pqueue->rear = pqueue->rear->next;

*//pqueue->rear = temp;*

++pqueue->size;

}

}

void QueuePop(Queue \*pqueue)

{

assert(pqueue != NULL);

Node \*del = NULL;

assert(pqueue != NULL);

if (pqueue->front == NULL) {

return;

}

del = pqueue->front;

pqueue->front = pqueue->front->next;

free(del);

del = NULL;

++pqueue->size;

}

Node \*QueueTop(Queue \*pqueue)

{

return pqueue->front;

}

bool QueueEmpty(const Queue \*pqueue)

{

return pqueue->front == NULL;

}

size\_t QueueSize(const Queue \*pqueue)

{

return pqueue->size;

}

main函数:

#include <stdio.h>

#include <unistd.h>

#include <assert.h>

#include <string.h>

#include <stdlib.h>

#include <stdbool.h>

#include "head.h"

void RR(Queue \*pqueue, PCB pcbs[], int size)

{

*//这里是时间片的大小*

int q = 0;

printf("please input q:");

scanf("%d", &q);

*//这个表示本次时间片q剩下多少*

int lessq = 0;

int pcbnumber = size;

*//表示本进程还需要的服务时间*

int time = 0;

*//*

*// |\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\_\_\_\_\_\_\_|\_\_\_*

*// 0 1 2 3 4 5 6 7 8*

while (1) {

*//等待进程*

for (int i = 0; i < size; i++) {

if (pcbs[i].ArrivalTime == time) {

QueuePush(pqueue, pcbs[i]);

printf("QueueTop(pqueue)->next==%p\n", QueueTop(pqueue)->next);

printf("PCB %c ARRIVAL TIME IS %d\n", pcbs[i].name, time);

pcbnumber--;

*// printf("pcbnumber == %d\n", pcbnumber);*

}

}

if (QueueEmpty(pqueue)&&pcbnumber ==0) {

printf("进程执行完毕！\n");

break;

}

*// printf("time is %d \n", time);*

*//*

*//*

*//执行进程*

if (lessq == 0) {

if (!QueueEmpty(pqueue) && QueueTop(pqueue)->data.ResidualTime== 0) {

printf("PCB %c FINISH TIME IS %d\n", QueueTop(pqueue)->data.name, time);

QueuePop(pqueue);

}

else {

if (!QueueEmpty(pqueue) &&

(QueueTop(pqueue)->data.ServiceTime !=

QueueTop(pqueue)->data.ResidualTime)) {

pqueue->rear->next = pqueue->front;

pqueue->rear = pqueue->rear->next;

pqueue->front = pqueue->front->next;

pqueue->rear->next = NULL;

}

}

lessq = q;

}else {

if (!QueueEmpty(pqueue) && QueueTop(pqueue)->data.ResidualTime == 0) {

printf("PCB %c FINISH TIME IS %d\n", QueueTop(pqueue)->data.name, time);

QueuePop(pqueue);

lessq = q;

}

}

if (!QueueEmpty(pqueue)) {

QueueTop(pqueue)->data.ResidualTime--;

}

lessq--;

time++;

*//sleep(1);*

}

}

int main()

{

int processnumber = 0;

Queue queue;

linkList linklist;

printf("输入进程数量: ");

scanf("%d", &processnumber);

*// 分配内存*

PCB \*pcbs = (PCB \*) malloc (sizeof(PCB) \* processnumber);

memset(pcbs, 0, sizeof(PCB) \* processnumber);

for (int i = 0; i < processnumber; i++) {

printf("进程名: ");

scanf(" %c",&(pcbs[i].name));

printf("到达时间: ");

scanf("%d",&(pcbs[i].ArrivalTime));

printf("服务时间: ");

scanf("%d",&(pcbs[i].ServiceTime));

pcbs[i].ResidualTime = pcbs[i].ServiceTime;

printf("优先级: ");

scanf("%d",&(pcbs[i].Priority));

}

int input = 0;

QueueInit(&queue);

RR(&queue, pcbs, processnumber);

QueueDestroy(&queue);

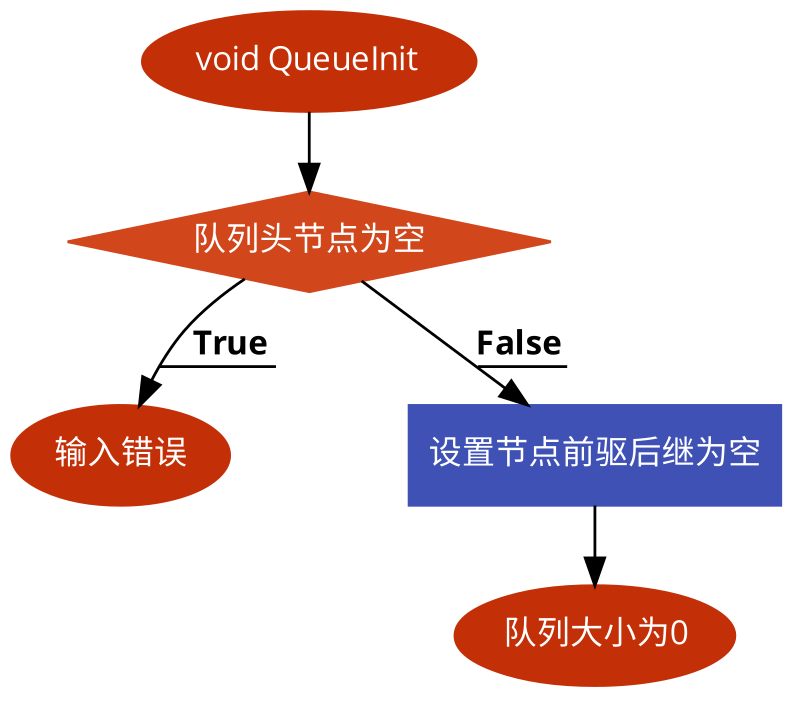
return 0;

}

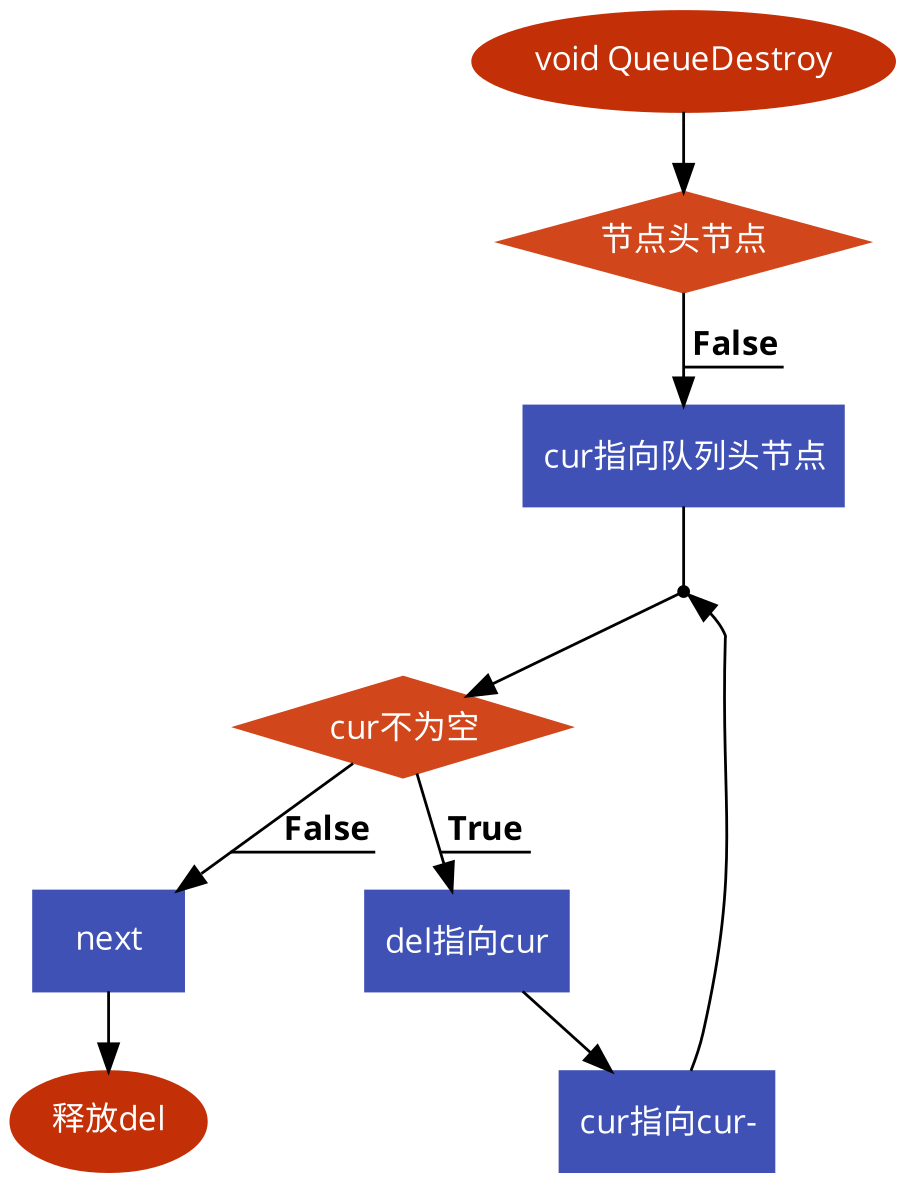
## 程序流程图:

数据结构:

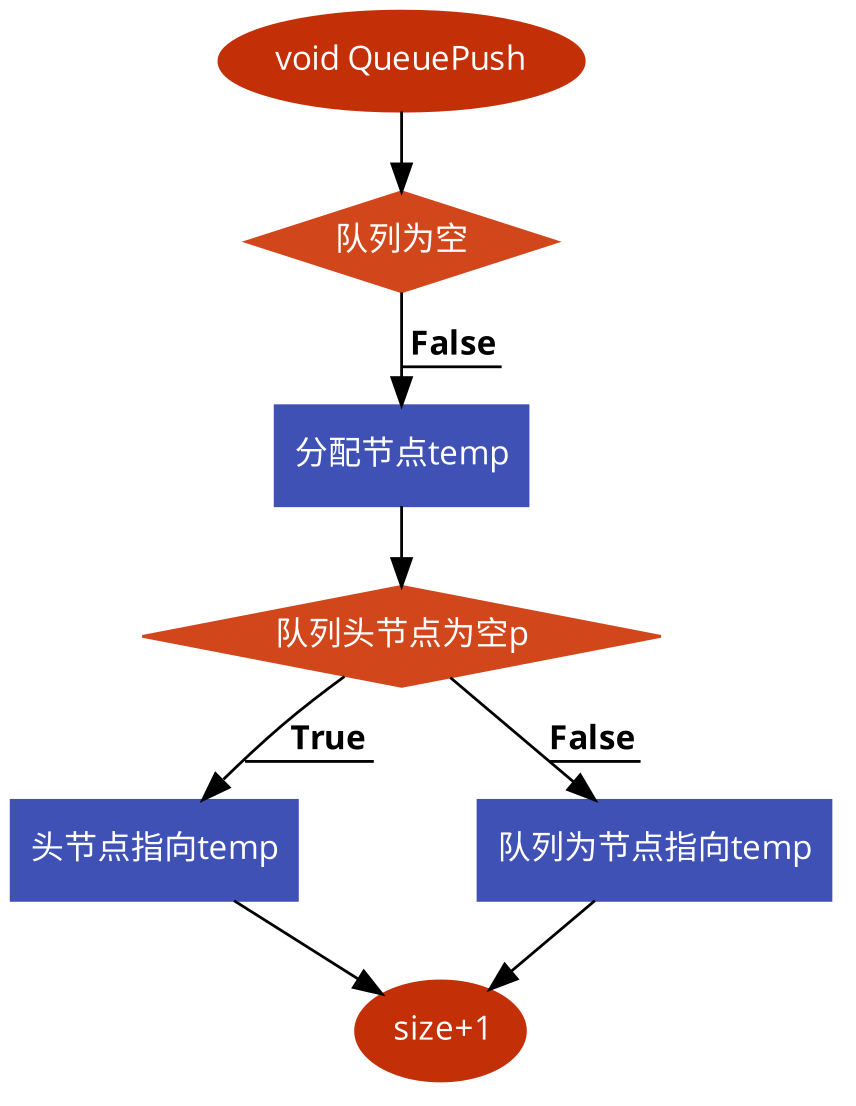
QueueInit



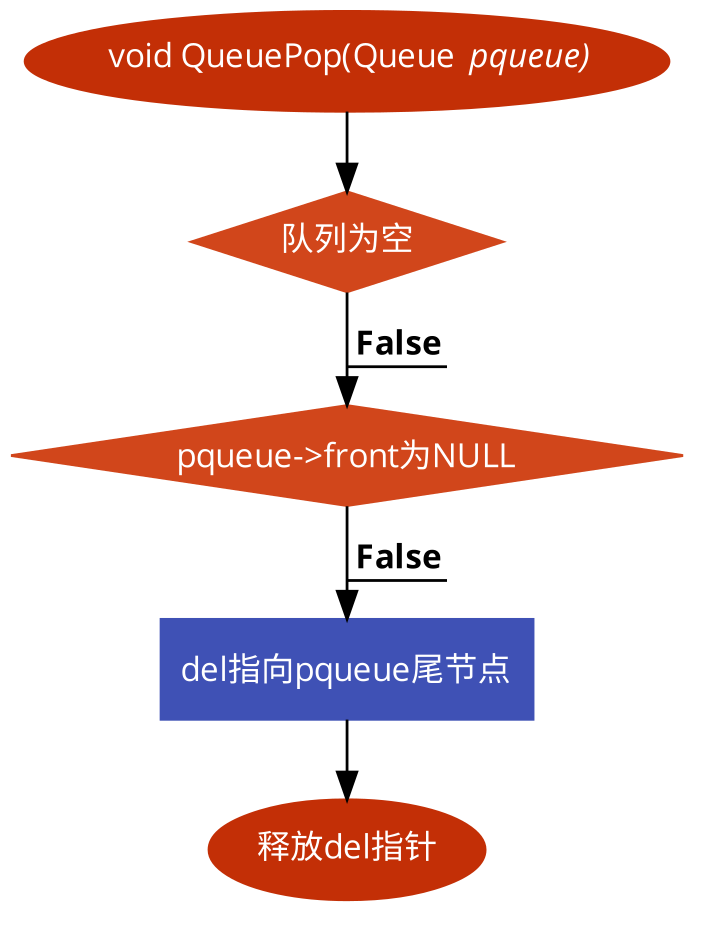
QueueDestroy



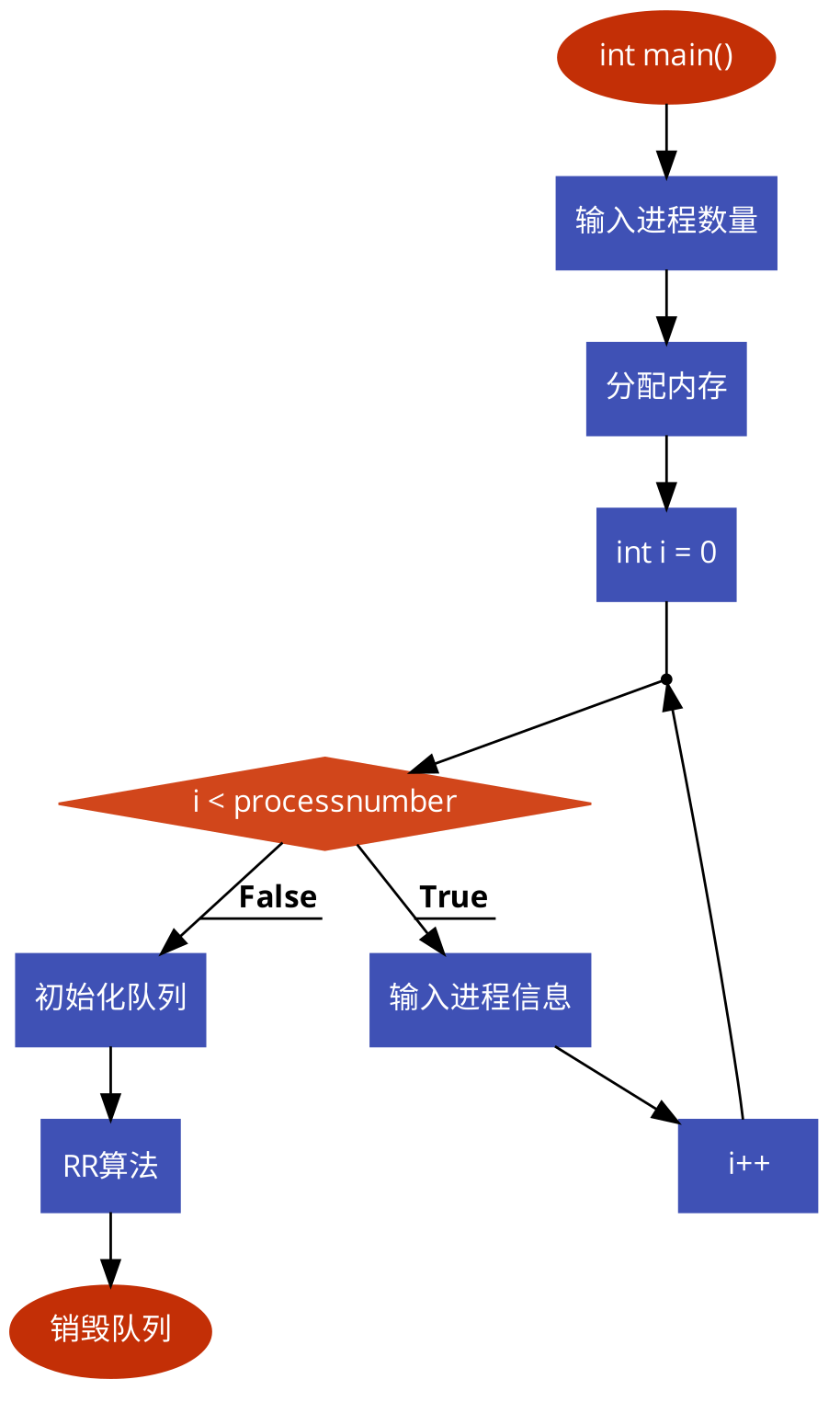
QueuePush



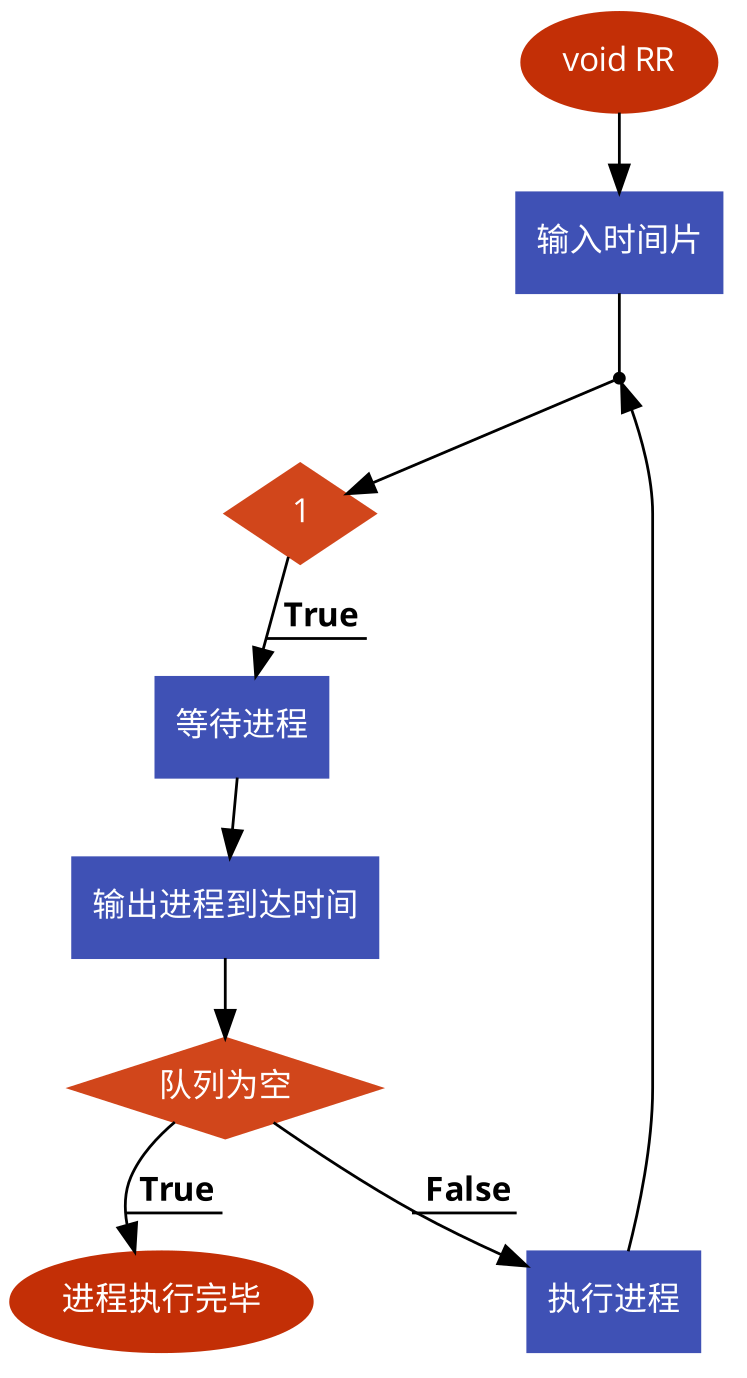
## QueuePop



## 主函数



## RR算法:



## 运行结果:

