**信息科学技术学院**

**课程（实习）设计**

**课程设计名称：**

**专 业：**

**学 号：**

**学 生 姓 名：**

**成 绩：**

**批 改 日 期：**

**教 师 签 名：**

目 录

题目（一级标题，16磅，宋体，居中） 1

1.1 设计目的（二级标题，小四，宋体） 1

1.2 实验预备内容 1

1.3设计内容 1

1.4实现代码 1

1.5结果展示 1

1.6总结 1

# 题目（一级标题，16磅，宋体，居中）

## 1.1 设计目的（二级标题，小四，宋体）

掌握FIFO，OPT，LRU等页面置换算法

## 1.2 实验预备内容

来自任务书内容

## 1.3设计内容

来自任务书内容

……

## 1.4实现代码

关键代码如下:

OPT算法

int opt()

{

int pg[400];

// 生成随机数种子

srand((int)time(0) \* 1000);

// 填充

for (int i = 0; i < 400; i++) {

pg[i] = rand() % 10;

}

int pn = sizeof(pg) / sizeof(pg[0]);

int fn = 10;

// 调用opt算法

optimalPage(pg, pn, fn);

return 0;

}

int predict(int pg[], vector<int>& fr, int pn, int index)

{

// 设置最远访问为index

int res = -1, farthest = index;

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

int j;

for (j = index; j < pn; j++) {

if (fr[i] == pg[j]) {

// 更新最远信息

if (j > farthest) {

farthest = j;

res = i;

}

break;

}

}

if (j == pn)

return i;

}

return (res == -1) ? 0 : res;

}

void optimalPage(int pg[], int pn, int fn)

{

// 容器

vector<int> fr;

int hit = 0;

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

// pg里面搜索fr

if (search(pg[i], fr)) {

// 命中

hit++;

continue;

}

// 数量小于fn

if (fr.size() < fn)

fr.push\_back(pg[i]);

else {

// 预测

int j = predict(pg, fr, pn, i + 1);

fr[j] = pg[i];

}

}

printf("No. of hits = %d\n",hit);

printf("No. of misses = %d\n", pn - hit);

}

bool search(int key, vector<int>& fr)

{

// 遍历找到key

for (int i = 0; i < fr.size(); i++)

if (fr[i] == key)

return true;

return false;

}

先进先出算法:

int fifo()

{

int referenceString[400];

int pageFaults = 0, m, n, s, pages, frames;

pages = 40;

// 生成种子

srand((int)time(0) \* 1000);

for(m = 0; m < pages; m++) {

// 赋值指令

referenceString[m] = rand() % 10;

}

frames = 10;

int temp[frames];

// 初始化

for(m = 0; m < frames; m++)

{

temp[m] = -1;

}

// 遍历所有页

for(m = 0; m < pages; m++)

{

s = 0;

// 遍历所有帧

for(n = 0; n < frames; n++)

{

// 命中

if(referenceString[m] == temp[n])

{

s++;

pageFaults--;

}

}

pageFaults++;

// 更新

if((pageFaults <= frames) && (s == 0))

{

temp[m] = referenceString[m];

}

else if(s == 0)

{

temp[(pageFaults - 1) % frames] = referenceString[m];

}

printf("\n");

for(n = 0; n < frames; n++)

{

printf("%d\t", temp[n]);

}

}

printf("\nTotal Page Faults:\t%d\n", pageFaults);

return 0;

}

LRU：

int lru()

{

int no\_of\_frames, no\_of\_pages, frames[10], pages[40], counter = 0, time[10], flag1, flag2, i, j, pos, faults = 0;

// 初始化

no\_of\_frames = 10;

no\_of\_pages = 40;

// srand(time(0) \* 1000);

// 填充指令

for(i = 0; i < no\_of\_pages; ++i){

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

pages[i] = rand() % 10;

}

// 初始化

for(i = 0; i < no\_of\_frames; ++i){

frames[i] = -1;

}

for(i = 0; i < no\_of\_pages; ++i){

flag1 = flag2 = 0;

for(j = 0; j < no\_of\_frames; ++j){

// 命中

if(frames[j] == pages[i]){

counter++;

// 设置计数器

time[j] = counter;

// 设置flalg

flag1 = flag2 = 1;

break;

}

}

// 检查flag

if(flag1 == 0){

for(j = 0; j < no\_of\_frames; ++j){

if(frames[j] == -1){

counter++;

faults++;

frames[j] = pages[i];

time[j] = counter;

flag2 = 1;

break;

}

}

}

if(flag2 == 0){

pos = min\_time\_pos(time, no\_of\_frames);

counter++;

faults++;

frames[pos] = pages[i];

time[pos] = counter;

}

printf("\n");

for(j = 0; j < no\_of\_frames; ++j){

printf("%d\t", frames[j]);

}

}

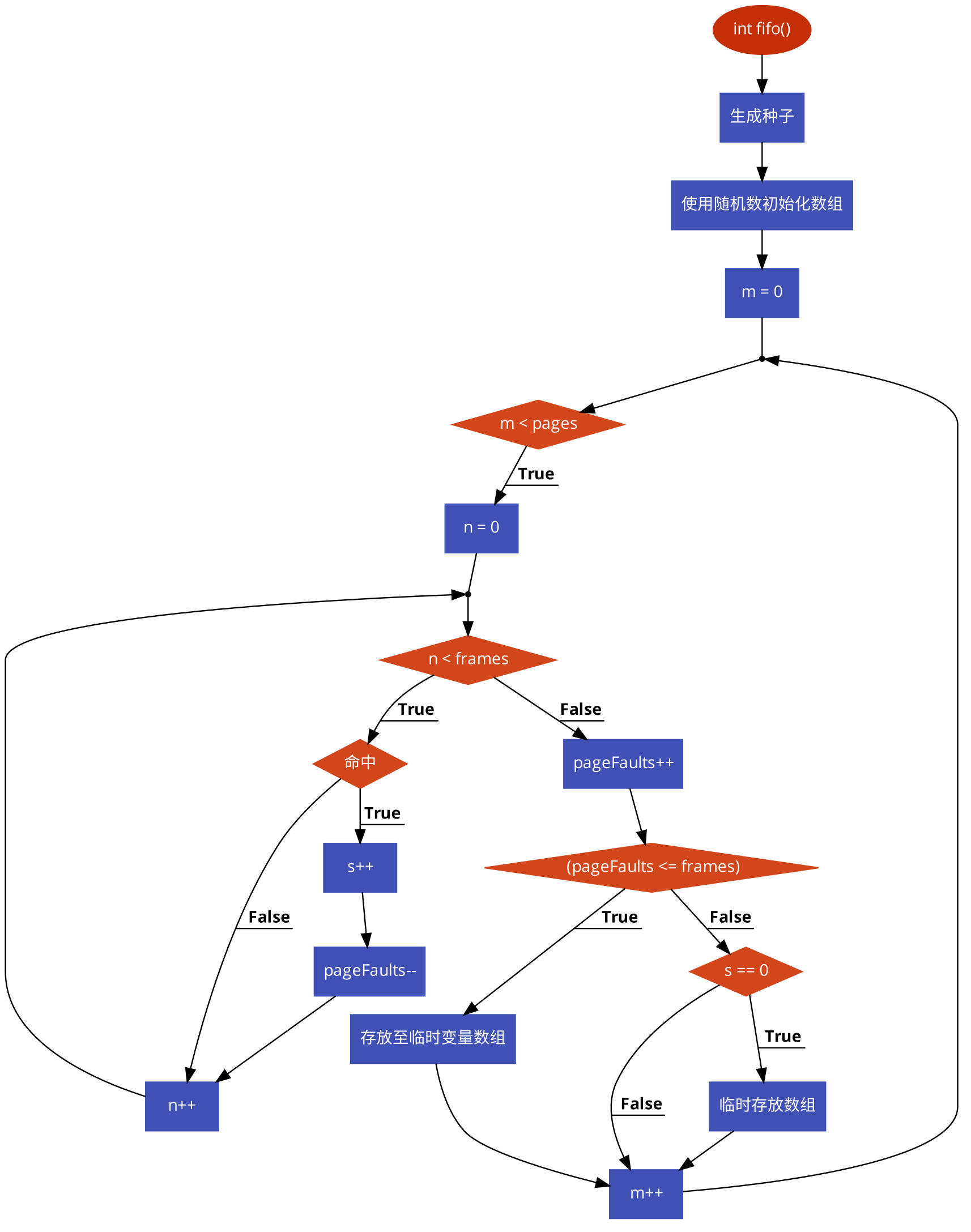
printf("\n\nTotal Page Faults = %d", faults);

return 0;

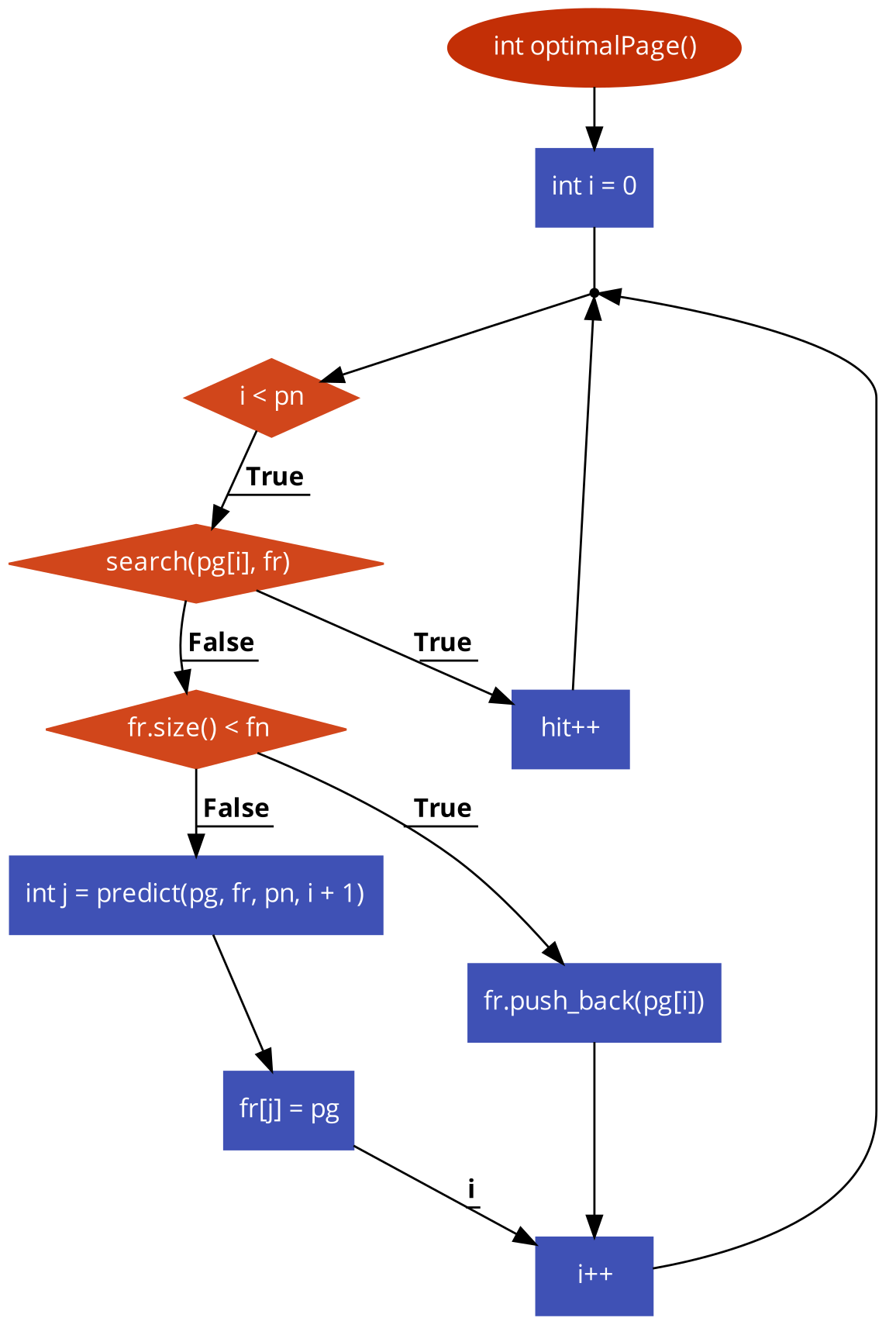
}

程序流程图如下:

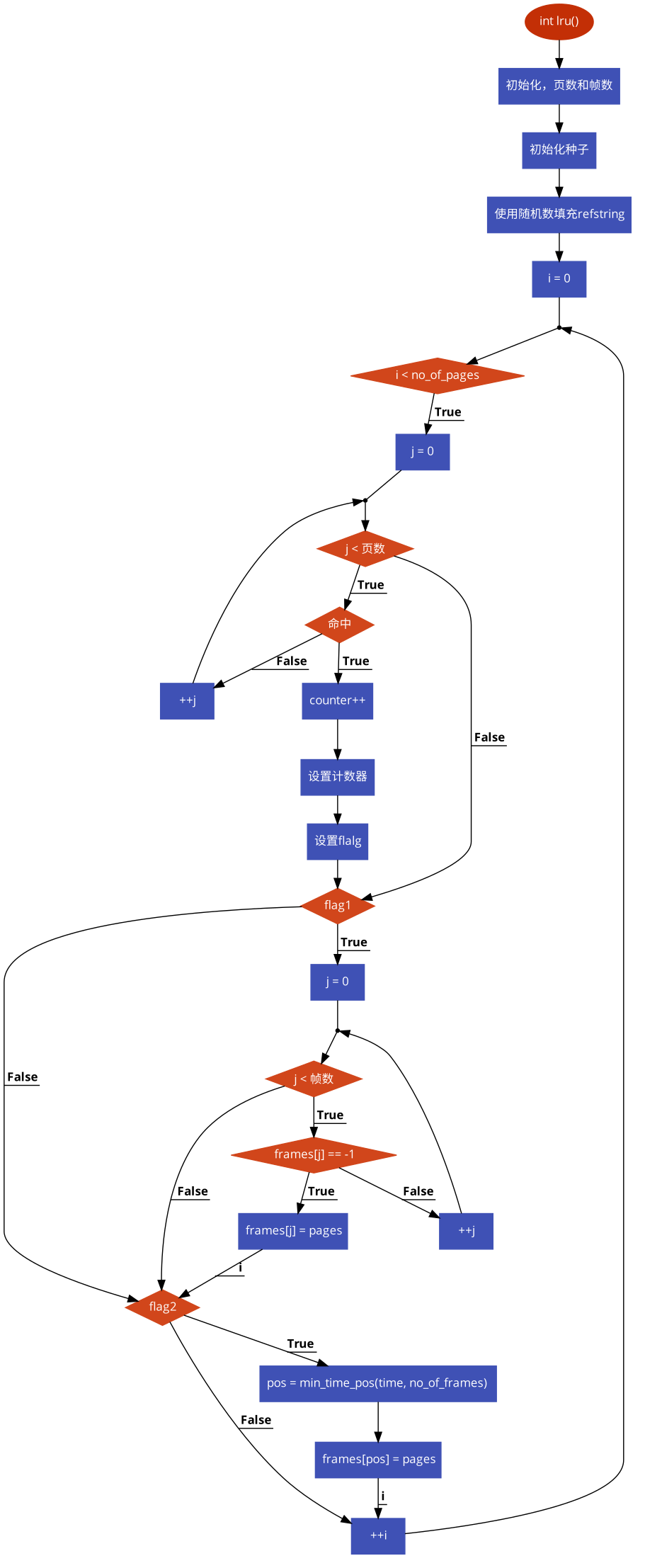
FIFO:



OPT:

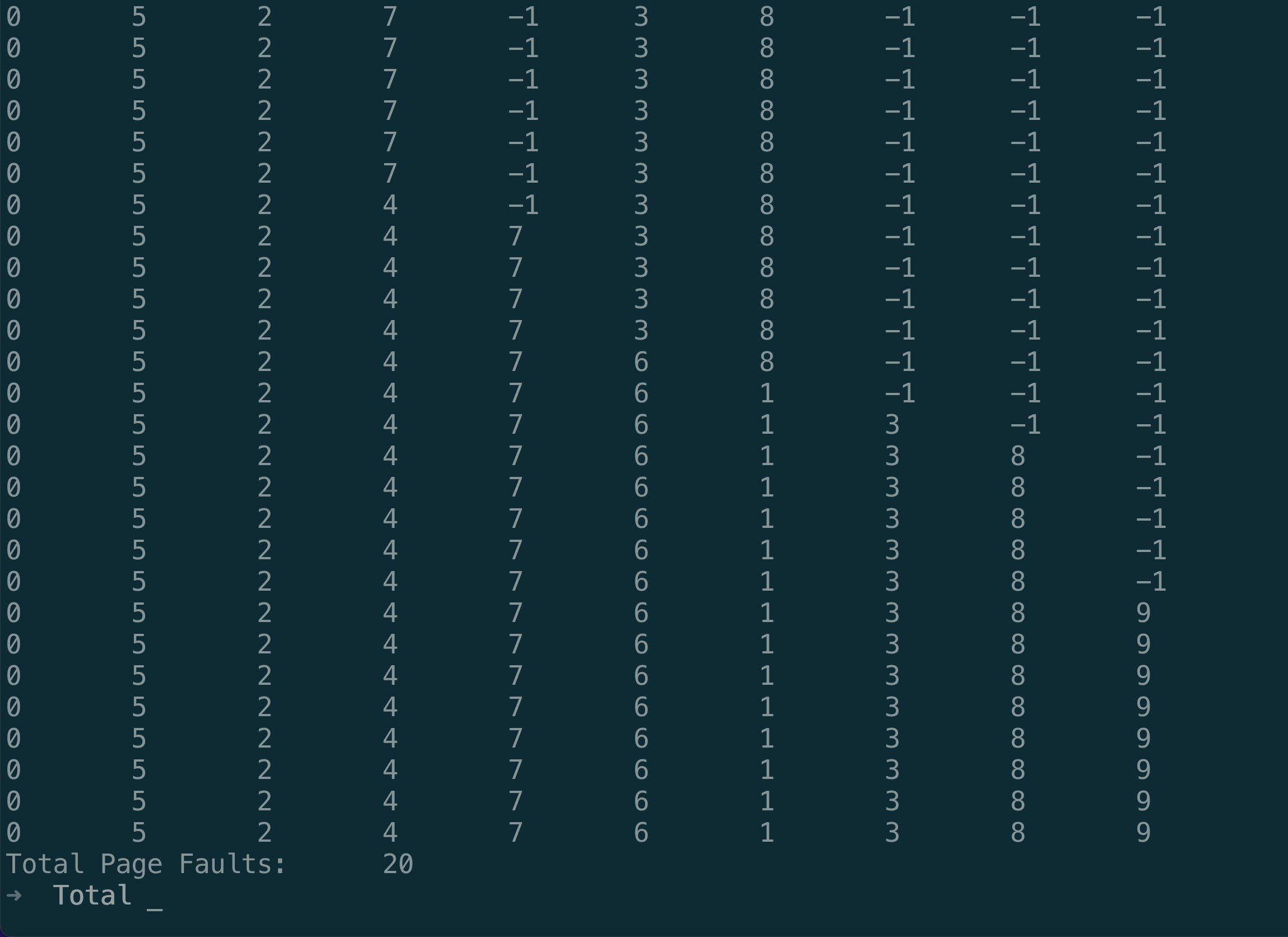


LRU:

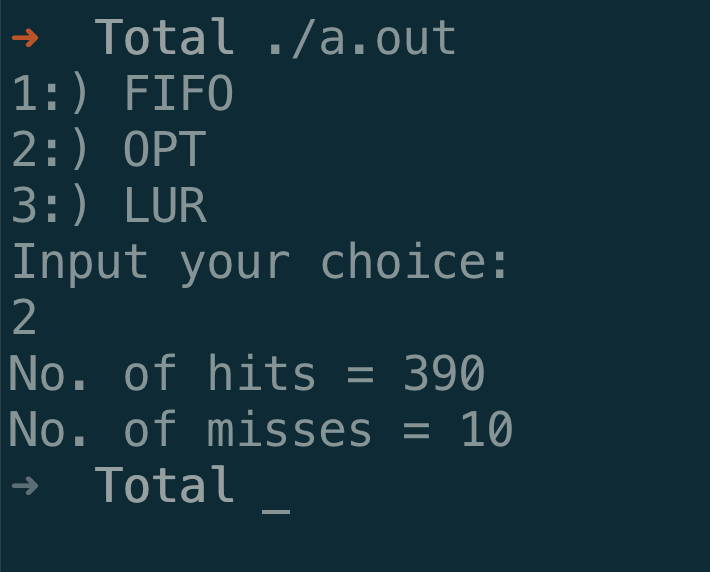


## 1.5结果展示

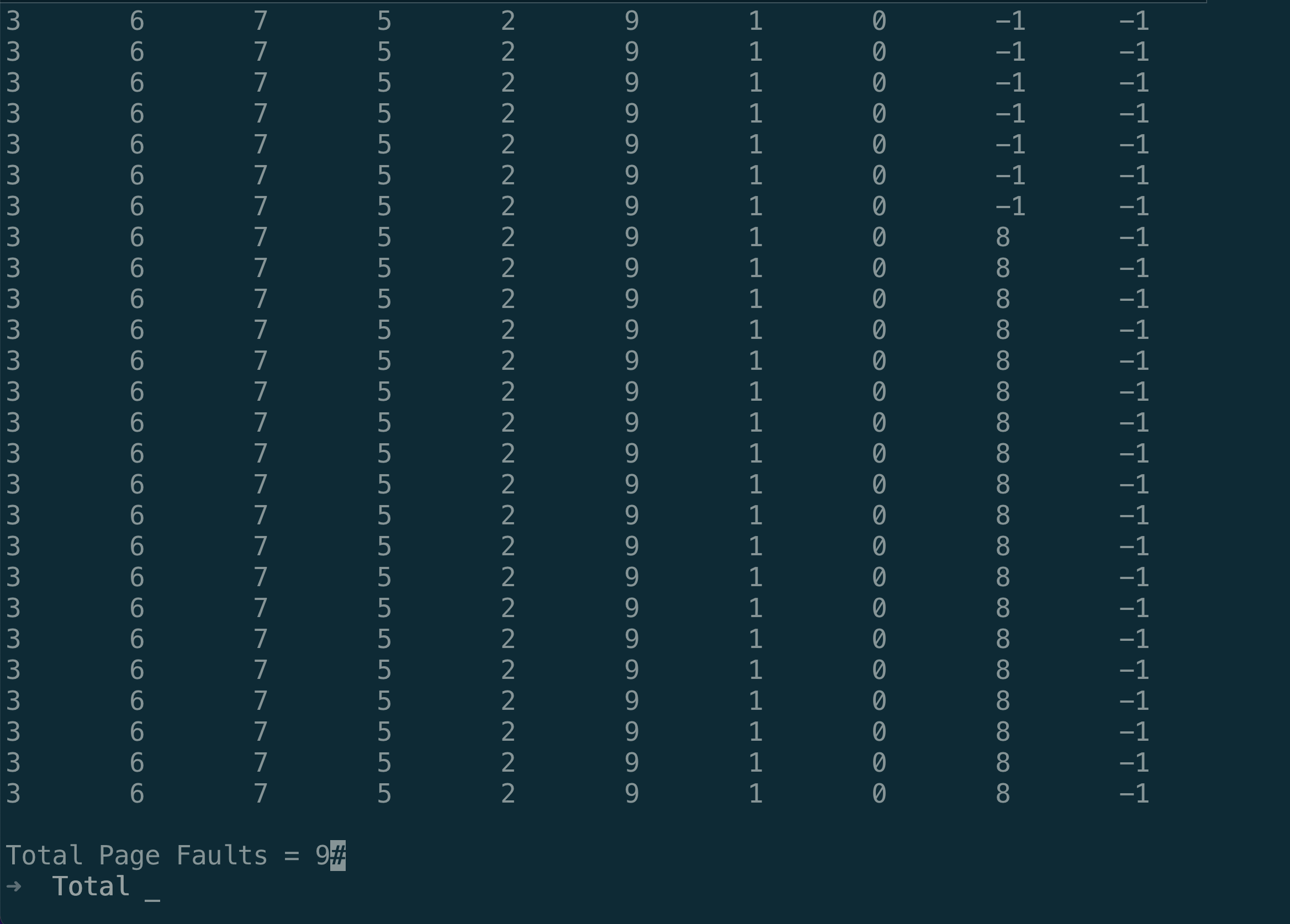
FIFO算法



OPT算法



LRU算法:



## 1.6总结

通过本次实验学了到了，操作系统中FIFO，OPT，和LRU页面置换算法的简单实现，通过使用C/C++语言进行了简单的编写和实现。在使用rand（）函数生成随机数字的时候需要使用srand进行初始化，使用time（）函数返回的值作为其参数值，否则每次生成的随机数都是一样的。学习到了Linux操作系统的使用方法，使用vim编写C/C++程序，使用Gcc或者g++编译程序，使用./a.out执行程序。