课题二参考：

Simulate the detail operation for Belady’s Anomaly**,** including**:**

1. Algorithm flowchart;

2. Code;

3. Output result.

4. A complete report.

Algorithm flowchart：

设置总的页面数

输入依次要访问的页面

分配内存物理块数量

物理内存中不存在该页面，添加该页面

物理内存中已存在该页面，则不做处理，内存结构不改变

如果物理内存块已满，则替换掉最早进行操作的页面

如果物理内存块未满，则直接添加

每次访问页面，记录内存块中最早操作的块，继续添加要访问的页面

## Code：

#include <iostream>

#include <cstdlib>

#include <vector>

#include <cstdio>

#define pause system("pause")

using namespace std;

const int maxn = 1005;

int a[maxn];

int b[maxn];

int top;

void init(int m)

{

top = 0;

for (int i = 0; i < m; i++) b[i] = -1;

}

int insert(int val, int m)

{

int flag = 1;

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

{

if (b[i] == val)

{

flag = 0; //已经存在该值

break;

}

if (b[i] == -1)

{

flag = 1;

b[i] = val;

break;

}

}

if (flag)

{

b[top] = val;

}

top = (top + 1) % m;

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

{

cout << b[i] << " ";

}

if(flag == 1)

cout << " 缺页次数加1";

cout << endl;

return flag;

}

int main()

{

int n;

cout << "输入总访问页面数：";

cin >> n;

cout << "输入依次要访问的页面序列：";

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

cin >> a[i];

int m;

cout << "输入分配给该作业的页数：";

cin >> m;

init(m);

int cnt = 0;

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

{

cnt += insert(a[i], m);

}

cout << " 总缺页次数为："<< cnt << endl;

return 0;

}

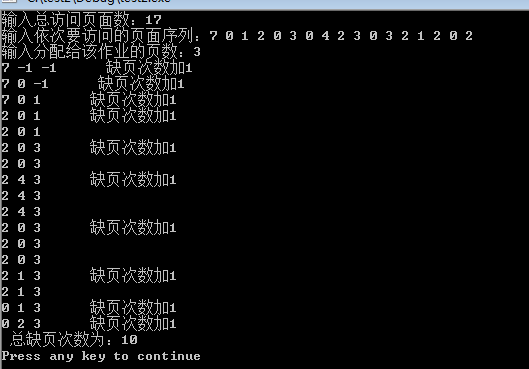
## Test Result：

页面总数为17

依次访问的页码为7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 2

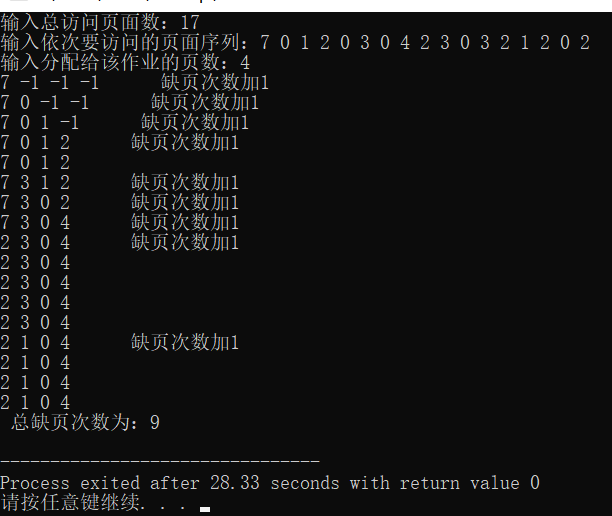
设置分配的页面数量为 3

这时的缺页次数为10



设置分配的页面数量为4时

这时的缺页次数为9



FIFO算法即将程序要运行的页码放入物理内存块，当内存块空间满了的时候，就将最早进行操作的页码替换掉，加入新的需要的页码，这也就是所谓的缺页。如果访问的页码已经存在物理内存块中，就不用进行操作，但是要更新最早进行过操作的页面。

访问相同序列的页码，可以发现页面数小的缺页次数要大于页面数大的。即如果内存中页数更小的集合是内存页数更大的集合的子集，这个算法被称为stack algorithm。可以证明stack algorithm(如LRU)不会出现belady现象，FIFO会出现。