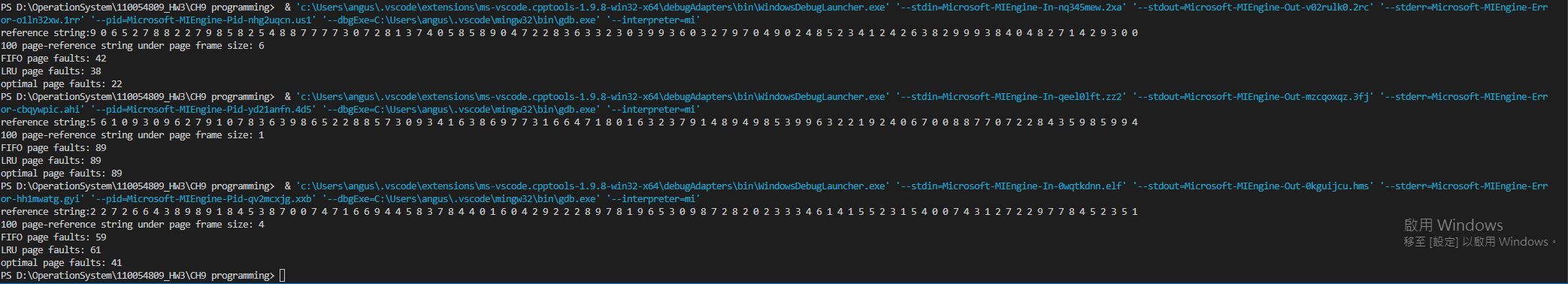
Snapshot：



Source code：

#include<stdio.h>

#include<stdlib.h>

#include<time.h>

#include<stdbool.h>

#define page\_reference\_length 100

int \*referenceString;

int pageFrameSize;

bool page\_found(int searchPage, int pageList[]) {

for (size\_t i = 0; i < pageFrameSize; i++) {

if (searchPage == pageList[i]) {

return true;

}

}

return false;

}

int FIFO(int pageList[]) {

int pageFaults = 0, j = 0;;

for (size\_t i = 0; i < page\_reference\_length; i++) {

if (!page\_found(referenceString[i], pageList)) {

pageFaults++;

for (j = 0; j < pageFrameSize-1; j++) {

pageList[j] = pageList[j+1];

}

pageList[j] = referenceString[i];

}

}

return pageFaults;

}

int findLastUseIndex(int curIndex, int target) {

for (curIndex--; curIndex >= 0; curIndex--) {

if (referenceString[curIndex] == target) {

return curIndex;

}

}

return 0;

}

int LRU(int pageList[]) {

int pageFaults = 0;

int blockFull = 0, curFillIndex = 0;

for (size\_t i = 0; i < page\_reference\_length; i++) {

if (!page\_found(referenceString[i], pageList)) {

pageFaults++;

if (curFillIndex == pageFrameSize) {

blockFull = 1;

}

if (blockFull == 0) {

pageList[curFillIndex] = referenceString[i];

curFillIndex++;

continue;

}

int lastApperence[pageFrameSize];

for (size\_t j = 0; j < pageFrameSize; j++) {

lastApperence[j] = findLastUseIndex(i, pageList[j]);

}

int farest = lastApperence[0], replaceIndex = 0;

for (size\_t j = 1; j < pageFrameSize; j++) {

if (lastApperence[j] < farest) {

farest = lastApperence[j];

replaceIndex = j;

}

}

pageList[replaceIndex] = referenceString[i];

}

}

return pageFaults;

}

int findNextIndex(int curIndex, int target) {

for (curIndex++; curIndex < page\_reference\_length; curIndex++) {

if (referenceString[curIndex] == target) {

return curIndex;

}

}

return page\_reference\_length;

}

int optimal(int pageList[]) {

int pageFaults = 0;

int blockFull = 0, curFillIndex = 0;

for (size\_t i = 0; i < page\_reference\_length; i++) {

if (!page\_found(referenceString[i], pageList)) {

pageFaults++;

if (curFillIndex == pageFrameSize) {

blockFull = 1;

}

if (blockFull == 0) {

pageList[curFillIndex] = referenceString[i];

curFillIndex++;

continue;

}

int nextApperence[pageFrameSize];

for (size\_t j = 0; j < pageFrameSize; j++) {

nextApperence[j] = findNextIndex(i, pageList[j]);

}

int farest = nextApperence[0], replaceIndex = 0;

for (size\_t j = 1; j < pageFrameSize; j++) {

if (nextApperence[j] > farest) {

farest = nextApperence[j];

replaceIndex = j;

}

}

pageList[replaceIndex] = referenceString[i];

}

}

return pageFaults;

}

void generatePageStrings() {

// referenceString = (int \*)malloc(sizeof(int) \* page\_reference\_length);

// referenceString[0] = 2;

// referenceString[1] = 7;

// referenceString[2] = 7;

// referenceString[3] = 2;

// referenceString[4] = 4;

// referenceString[5] = 7;

// referenceString[6] = 0;

// referenceString[7] = 5;

// referenceString[8] = 7;

// referenceString[9] = 5;

srand(time(0));

referenceString = (int \*)malloc(sizeof(int) \* page\_reference\_length);

printf("reference string:");

for (size\_t i = 0; i < page\_reference\_length; i++) {

referenceString[i] = rand() % 10;

printf("%d ", referenceString[i]);

}

printf("\n");

return;

}

int generatePageFrame() {

srand(time(0));

int pageFrame;

pageFrame = rand() % 7 + 1;

return pageFrame;

}

void resetPageList(int pageList[]) {

for (size\_t i = 0; i < pageFrameSize; i++) {

pageList[i] = -1;

}

return;

}

int main(int argc, char\* argv[]) {

pageFrameSize = generatePageFrame();

int pageList[pageFrameSize];

resetPageList(pageList);

generatePageStrings();

int FIFO\_page\_faults, LRU\_page\_faults, optimal\_page\_faults;

FIFO\_page\_faults = FIFO(pageList);

resetPageList(pageList);

LRU\_page\_faults = LRU(pageList);

resetPageList(pageList);

optimal\_page\_faults = optimal(pageList);

printf("100 page-reference string under page frame size: %d\n", pageFrameSize);

printf("FIFO page faults: %d\n", FIFO\_page\_faults);

printf("LRU page faults: %d\n", LRU\_page\_faults);

printf("optimal page faults: %d\n", optimal\_page\_faults);

free(referenceString);

return 0;

}