1. FIFO Paging

#include <stdio.h>

#define MAX\_FRAMES 10

#define MAX\_PAGES 100

void fifoPageReplacement(int pages[], int n, int frames) {

int pageFrame[MAX\_FRAMES], front = 0, pageFaults = 0;

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

pageFrame[i] = -1;

}

printf("Page Reference String: ");

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

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

}

printf("\n\n");

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

int found = 0;

for (int j = 0; j < frames; j++) {

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

found = 1;

break;

}

}

if (!found) {

pageFrame[front] = pages[i];

front = (front + 1) % frames;

pageFaults++;

printf("Page %d caused a fault. Frames: ", pages[i]);

for (int j = 0; j < frames; j++) {

if (pageFrame[j] == -1)

printf(" - ");

else

printf("%d ", pageFrame[j]);

}

printf("\n");

} else {

printf("Page %d is already in the frame (Hit).\n", pages[i]);

}

}

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

}

int main() {

int n, frames;

int pages[MAX\_PAGES];

printf("Enter number of pages: ");

scanf("%d", &n);

printf("Enter the page reference string: ");

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

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

}

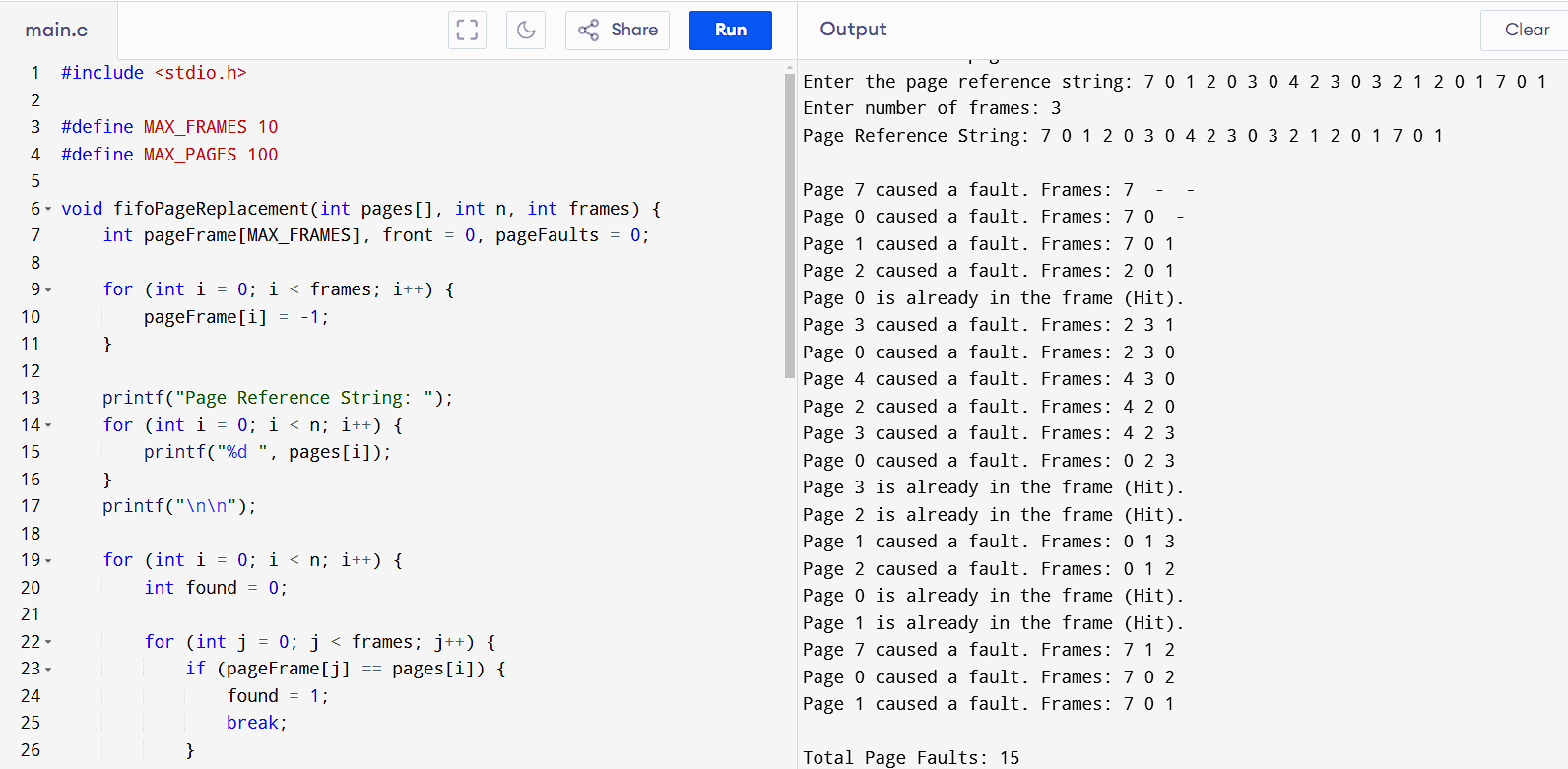
printf("Enter number of frames: ");

scanf("%d", &frames);

fifoPageReplacement(pages, n, frames);

return 0;

}



1. LRU Paging

#include <stdio.h>

#define MAX\_FRAMES 10

#define MAX\_PAGES 100

int findLRU(int time[], int frames) {

int min = time[0], pos = 0;

for (int i = 1; i < frames; i++) {

if (time[i] < min) {

min = time[i];

pos = i;

}

}

return pos;

}

void lruPageReplacement(int pages[], int n, int frames) {

int pageFrame[MAX\_FRAMES], time[MAX\_FRAMES], pageFaults = 0, counter = 0;

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

pageFrame[i] = -1;

}

printf("Page Reference String: ");

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

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

}

printf("\n\n");

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

int found = 0;

for (int j = 0; j < frames; j++) {

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

found = 1;

time[j] = counter++;

break;

}

}

if (!found) {

int pos;

if (i < frames) {

pos = i;

} else {

pos = findLRU(time, frames);

}

pageFrame[pos] = pages[i];

time[pos] = counter++;

pageFaults++;

printf("Page %d caused a fault. Frames: ", pages[i]);

for (int j = 0; j < frames; j++) {

if (pageFrame[j] == -1)

printf(" - ");

else

printf("%d ", pageFrame[j]);

}

printf("\n");

} else {

printf("Page %d is already in the frame (Hit).\n", pages[i]);

}

}

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

}

int main() {

int n, frames;

int pages[MAX\_PAGES];

printf("Enter number of pages: ");

scanf("%d", &n);

printf("Enter the page reference string: ");

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

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

}

printf("Enter number of frames: ");

scanf("%d", &frames);

lruPageReplacement(pages, n, frames);

return 0;

}

4. Sequential file allocation

#include <stdio.h>

#define MAX\_FILES 100

struct File {

int start;

int length;

};

int disk[MAX\_FILES] = {0}; // 0 means free, 1 means occupied

struct File files[MAX\_FILES];

int file\_count = 0;

void allocateFile(int start, int length) {

if (start < 0 || start + length > MAX\_FILES) {

printf("Invalid allocation range!\n");

return;

}

for (int i = start; i < start + length; i++) {

if (disk[i] == 1) {

printf("Allocation failed! Block %d is already occupied.\n", i);

return;

}

}

for (int i = start; i < start + length; i++) {

disk[i] = 1;

}

files[file\_count].start = start;

files[file\_count].length = length;

file\_count++;

printf("File allocated from %d to %d\n", start, start + length - 1);

}

void displayDisk() {

printf("Disk Blocks:\n");

for (int i = 0; i < MAX\_FILES; i++) {

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

}

printf("\n");

}

int main() {

allocateFile(5, 10);

allocateFile(20, 5);

allocateFile(8, 6); // Overlapping allocation

displayDisk();

return 0;

}

