**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 10

дисциплина «Операционные системы»

«Алгоритмы замещения страниц.»

Выполнил: студент гр. ИТП–11

Ермолин Д.К.

Принял: преподаватель

Карась О.В.

Гомель 2022

**Цель работы:** разработать программу, реализующую заданный алгоритм замещения страниц в памяти.

**Задание:**

Вариант 11: Глобальное размещение. Алгоритм замещения – FIFO. Реализуется очередь страниц в конце которой попадают страницы, размещенные в памяти, а из начала берутся для замещения.

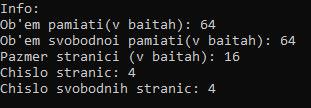


Рисунок 1 – Информация

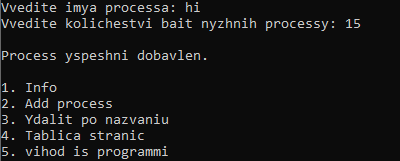


Рисунок 2 – Добавление первого процесса

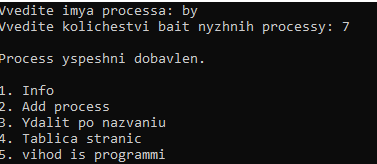


Рисунок 3 – Добавление второго процесса

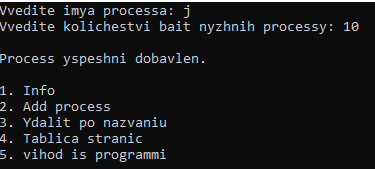


Рисунок 4 – Добавление третьего процесса

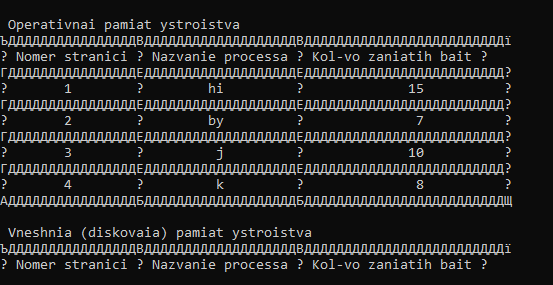


Рисунок 5 – Таблица страниц

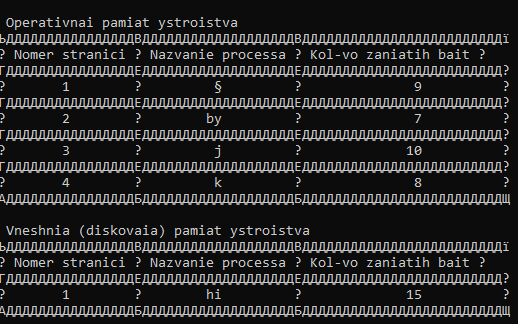


Рисунок 6 – Таблица страниц с пятым процессом

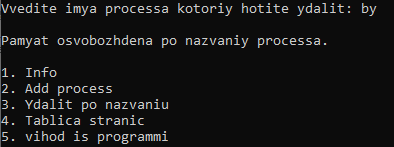


Рисунок 7 – Удаление процесса

**Вывод:** в данной лабораторной работе была разработана программа, реализующая заданный алгоритм замещения страниц в памяти.

**Приложение А**

**Листинг программы:**

#include <stdio.h>

#include <conio.h>

#include <string.h>

#include <stdlib.h>

#include <iostream>

#include <locale.h>

#include <iomanip>

#include <windows.h>

#include <locale.h>

using namespace std;

int checkInt(){

string s;

cin >> s;

int f = 0;

for (int i =0; i < s.length(); i++){

if ((int)s[i] < 48 || (int)s[i] > 57){

f = 1;

break;

}

}

while (f == 1){

f = 0;

cout << "Ne chislo ";

cin >> s;

for (int i =0; i < s.length(); i++){

if ((int)s[i] < 48 || (int)s[i] > 57){

f = 1;

break;

}

}

}

int ds = 0, dec = 1;

for (int i = s.length()-1; i >= 0; i--){

ds += ((int)s[i]-48)\*dec;

dec \*= 10;

}

return(ds);

}

void add\_proc(string \*processes, string \*Vprocesses, int \*numbers, int \*Vnumbers, char\* memory[], char\* Vmemory[], int \*Idx, int &min, int col\_pages, int &j, int &k, int &size, int page\_size, int &free\_pages)

{

setlocale(LC\_ALL, "Russian");

string m;

printf("Vvedite imya processa: ");

fflush(stdin);

getline(cin, m);

int i,n,h;

printf("Vvedite kolichestvo bait nyzhnih processy: ");

n = checkInt();

int pages = n / page\_size; // Количество страниц для процесса

if(n % page\_size != 0)

pages++;

if(size >= n)

{

i = 0;

while(n != 0)

{

if(processes[i] == "" && numbers[i] == 0)

{

if(free\_pages > 0) //Уменьшение количесвта свободных страниц на 1

free\_pages--;

processes[i] = m;

if(pages > 1)

{

numbers[i] = page\_size;

memory[i] = (char\*) malloc (page\_size);

size = size - page\_size;

n = n - page\_size;

Idx[i] = j+1;

j++;

}

else

{

numbers[i] = n;

size = size - page\_size;

memory[i] = (char\*) malloc (n);

n = 0;

Idx[i] = j+1;

j++;

min = Idx[0];

for(h=0;h<col\_pages;h++)

if(min > Idx[h] && Idx[h]>0) min = Idx[h];

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

printf("\nProcess yspeshni dobavlen.");

return;

}

pages--;

}

i++;

}

}

else

{

i = 0;

while(n != 0)

{

if(processes[i] == "" && numbers[i] == 0)

{

if(free\_pages > 0) //Уменьшение количесвта свободных страниц на 1

free\_pages--;

processes[i] = m;

if(pages > 1)

{

numbers[i] = page\_size;

memory[i] = (char\*) malloc (page\_size);

size = size - page\_size;

n = n - page\_size;

Idx[i] = j+1;

j++;

}

pages--;

}

else if(processes[i] != "" && numbers[i] != 0 && size == 0) //Уже size = 0!!!!!!!!

{

if(Idx[i] != -1 && Idx[i] == min) // ЗАМЕЩЕНИЕ

{

for(int g=0; g<k+1; g++)

{

if(Vprocesses[g] == "" && Vnumbers[g] == 0)

{

Vprocesses[g] = processes[i];

Vnumbers[g] = numbers[i];

Vmemory[g] = memory[i];

break;

//Vprocesses[k] = processes[i];

//Vnumbers[k] = numbers[i];

//Vmemory[k] = memory[i];

}

}

k++;

processes[i] = m;

if(pages > 1)

{

numbers[i] = page\_size;

memory[i] = (char\*) malloc (page\_size);

n = n - page\_size;

Idx[i] = -1;

}

else

{

numbers[i] = n;

memory[i] = (char\*) malloc (n);

n = 0;

Idx[i] = -1;

printf("\nProcess yspeshno dabavlen v pamyat c zamesheniem.");

//return;

}

min = Idx[0];

for(h=0;h<col\_pages;h++)

if(min > Idx[h] && Idx[h]>0) min = Idx[h];

pages--;

}

}

i++;

if(i>col\_pages-1) i = 0;

}

}

}

void delete\_process(char\* memory[], char\* Vmemory[], string \*processes, string \*Vprocesses, int \*numbers, int \*Vnumbers, int \*Idx, int &j, int col\_pages, int k, int &size, int page\_size, int &free\_pages)

{

setlocale(LC\_ALL, "Russian");

string m;

printf("Vvedite imya processa kotoriy hotite ydalit: ");

fflush(stdin);

getline(cin, m);

int i,d=0,q=0;

for(i = 0; i < col\_pages; i++)

{

if(processes[i] == m && numbers[i] != 0)

{

processes[i] = "";

numbers[i] = 0;

free(memory[i]);

free\_pages++;

size = size + page\_size;

Idx[i] = 0;

d++;

}

}

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

{

if(Vprocesses[i] == m && Vnumbers[i] != 0)

{

Vprocesses[i] = "";

Vnumbers[i] = 0;

free(Vmemory[i]);

}

}

if(d == 0)

printf("\nNazvanie ne deistvitelno!");

else

printf("\nPamyat osvobozhdena po nazvaniy processa.");

}

void menu()

{

setlocale(LC\_ALL, "Russian");

puts("\n1. Info");

puts("2. Add process");

puts("3. Ydalit po nazvaniu");

puts("4. Tablica stranic");

puts("5. vihod is programmi");

printf("\nViberite pynkt menu: ");

}

void table(int col\_pages, string \*processes, int \*numbers) //Вывод таблицы

{

setlocale(LC\_ALL, "Russian");

int i;

//Шапка

const char \*S;

printf("%c", 218);

for(i=0;i<16;i++) printf("%c", 196); printf("%c", 194); for(i=0;i<19;i++) printf("%c", 196); printf("%c", 194);

for(i=0;i<25;i++) printf("%c", 196); printf("%c\n", 191);

printf("%c Nomer stranici %c Nazvanie processa %c Kol-vo zaniatih bait %c\n", 179,179,179,179);

for(int j=0; j<col\_pages; j++)

{

if(j!=col\_pages-1)

{

printf("%c", 195); for(i=0;i<16;i++) printf("%c", 196); printf("%c", 197);

for(i=0;i<19;i++) printf("%c", 196); printf("%c", 197); for(i=0;i<25;i++) printf("%c", 196); printf("%c\n", 180);

S = processes[j].c\_str();

printf("%c%8d %c%10s %c%15d %c\n", 179, j+1, 179, S, 179, numbers[j], 179);

}

else

{

printf("%c", 195); for(i=0;i<16;i++) printf("%c", 196); printf("%c", 197);

for(i=0;i<19;i++) printf("%c", 196); printf("%c", 197); for(i=0;i<25;i++) printf("%c", 196); printf("%c\n", 180);

S = processes[j].c\_str();

printf("%c %d %c%10s %c%15d %c\n", 179, j+1, 179, S, 179, numbers[j], 179);

printf("%c", 192); for(i=0;i<16;i++) printf("%c", 196); printf("%c", 193);

for(i=0;i<19;i++) printf("%c", 196); printf("%c", 193); for(i=0;i<25;i++) printf("%c", 196); printf("%c", 217);

}

}

}

int main()

{

setlocale(LC\_ALL, "Russian");

int size = 0; //Размер общей памяти

int col\_pages = 0; //Количество страниц

int page\_size = 0; // Размер одной страницы

int free\_pages = 0; //Количество свободных страниц

int j = 0, k1 = 0, min;

printf("Vvedite ob'em pamiati v baitah': ");

//cout << "Input size of memory (bites): ";

while(size <= 0)

size = checkInt();;

printf("Vvedite kol-vo stranic: ");

while(col\_pages <= 0)

col\_pages = checkInt();; //Количество страниц

page\_size = size / col\_pages; // Размер одной страницы

free\_pages = col\_pages;

int size1 = size;

int Idx[col\_pages];

min = Idx[0];

string processes[col\_pages]; //Массив имен процессов

string Vprocesses[col\_pages];

int numbers[col\_pages]; //Массив количества байтов процессов

int Vnumbers[col\_pages];

for(int i = 0; i < col\_pages; i++)

{

processes[i] = "";

Vprocesses[i] = "";

numbers[i] = 0;

Vnumbers[i] = 0;

}

char\* memory[col\_pages]; //Массив памяти для процесса

char\* Vmemory[col\_pages];

int k;

while(true)

{

menu();

scanf("%d",&k);

system("cls");

switch(k)

{

case 1:

printf("Info:\n");

printf("Ob'em pamiati(v baitah): %d\n",size);

printf("Ob'em svobodnoi pamiati(v baitah): %d\n",size1);

printf("Pazmer stranici (v baitah): %d\n", page\_size);

printf("Chislo stranic: %d\n", col\_pages);

printf("Chislo svobodnih stranic: %d\n", free\_pages);

break;

case 2:

add\_proc(processes, Vprocesses, numbers, Vnumbers, memory, Vmemory, Idx, min, col\_pages, j, k1, size1, page\_size, free\_pages);

break;

case 3:

delete\_process(memory, Vmemory, processes, Vprocesses, numbers, Vnumbers, Idx, j, col\_pages, k1, size1, page\_size, free\_pages);

break;

case 4:

printf("\n Operativnai pamiat ystroistva\n");

table(col\_pages, processes, numbers);

//table(col\_pages, processes, Idx);

printf("\n\n Vneshnia (diskovaia) pamiat ystroistva\n");

table(k1, Vprocesses, Vnumbers);

break;

case 5:

exit(0);

break;

default:

puts("Vibran ne pravilni pynkt menu!");

}

printf("\n");

}

return 0;

}