**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

по дисциплине «Операционные системы»

Управление виртуальной памятью

Выполнил студент

группы ИТИ-11

Коновальчук Д.В.

Проверил преподаватель

Карась О.В.

Гомель 2022

Задание

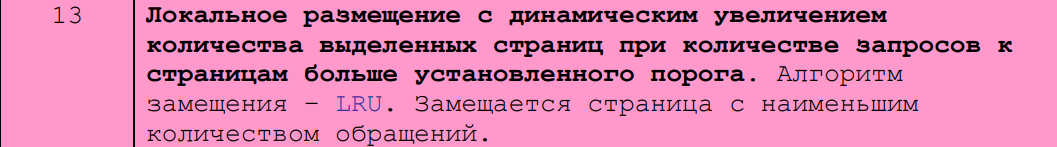
Разработать программу, реализующую заданный алгоритм замещения страниц в памяти. Менеджер памяти должен:

6. Разбивать память заданного размера на указанное количество страниц. На экран должна выводиться следующая информация о состоянии памяти: объем памяти, число страниц, число свободных страниц (%), размер страницы;

7. Размещать в памяти страницу заданного процесса, с замещением занятой по заданному алгоритму (по нажатию кнопки «ДОБАВИТЬ»). Для размещения страницы в памяти, указывается имя процесса и ее номер (вводятся отдельно). Например: Pro 3. После нажатия на кнопку «ДОБАВИТЬ» страница размещается в свободной странице памяти. Если задано глобальное размещение (см. вариант задания), то выбирается любая не занятая страница. При локальном размещении страница размещается только среди виртуальных страниц выделенных этому процессу. Выделение страниц в памяти выполняется при первом ее занесении процесса в память. Алгоритм замещения выполняется только при отсутствии свободных страниц под процесс.;

8. Удалять из памяти заданную страницу или все страницы заданного процесса (по нажатию кнопки «УДАЛИТЬ»). Указывается номер удаляемой страницы в памяти;

9. Организовывать циклическое обращение к страницам размещенным в памяти по нажатию на кнопку. При этом случайным образом задается количество обращений к страницам (диапазон 1..10). Для каждого обращения генерируется, случайным образом, номер страницы из диапазона [0; количество страниц памяти]. При обращении к странице в зависимости, от варианта, увеличивается ее внутренний счетчик обращений или устанавливается флаг обращения



Ход работы

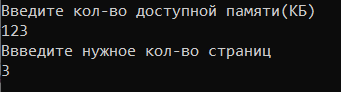


Рисунок 1 – Запуск программы



Рисунок 2 – Добавление процесса

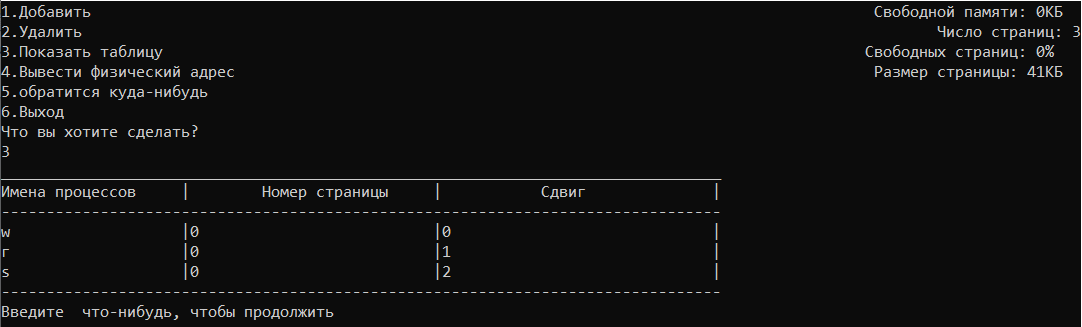


Рисунок 3 – Таблица процессов

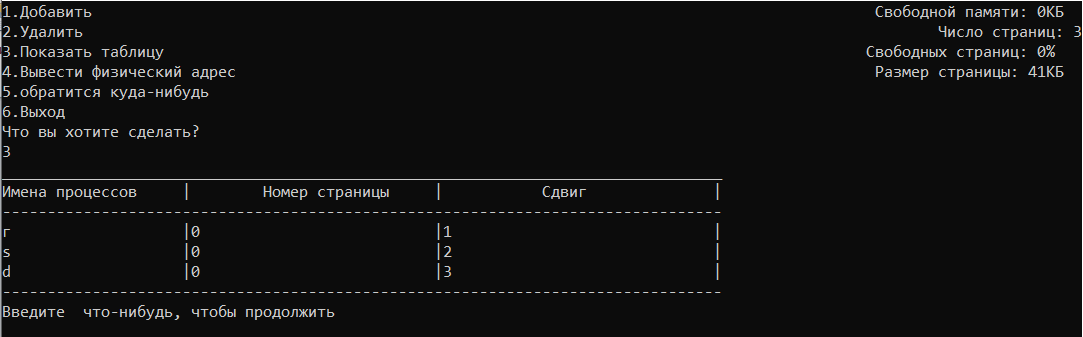


Рисунок 4 – Таблица процессов после добавления 4 процесса



Рисунок 5 – Удаление процесса

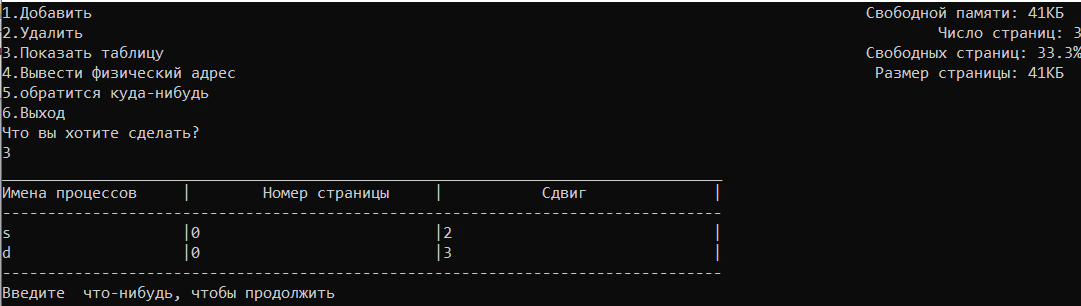


Рисунок 6 – Таблица процессов после удаления

Вывод: В ходе лабораторной работы изучили алгоритмы замещения страниц.

**ПРИЛОЖЕНИЕ А**

Листинг программы laba9.cpp

#include<iostream>

#include<stdlib.h>

int numLength(int n){

int s = 0;

if(n == 0)

return 1;

while(n > 0){

n = n / 10;

s++;

}

return s;

}

void makeSpaces(int n){

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

std::cout << " ";

}

struct data{

std::string name;

int page\_;

int padding\_;

int appeal; //обращений

};

void giveInfo(float available\_memory, int num\_of\_pages, int num\_of\_free\_pages, float size\_of\_page){

float percent\_of\_free\_pages;

int translate;

percent\_of\_free\_pages = (float)num\_of\_free\_pages \* 100 / (float)num\_of\_pages;

translate = percent\_of\_free\_pages \* 10;

percent\_of\_free\_pages = (float)translate / 10;

translate = size\_of\_page \* 10;

size\_of\_page = (float)translate / 10;

translate = available\_memory \* 10;

available\_memory = (float)translate / 10;

std::cout << "1.Добавить";

makeSpaces(88 - numLength(available\_memory));

std::cout << "Свободной памяти: " << available\_memory << "КБ\n2.Удалить";

makeSpaces(96 - numLength(num\_of\_pages));

std::cout << "Число страниц: " << num\_of\_pages << "\n3.Показать таблицу";

makeSpaces(79 - numLength(num\_of\_free\_pages));

std::cout << "Свободных страниц: " << percent\_of\_free\_pages << "%\n4.Вывести физический адрес";

makeSpaces(73 - numLength(size\_of\_page));

std::cout << "Размер страницы: " << size\_of\_page << "КБ\n5.обратится куда-нибудь\n6.Выход\n";

}

void showAddress(int page, int padding, data \*a, int prikol){

int i = 0;

std::string go\_next;

bool found = true;

while(i < prikol && found){

if((a+i)->page\_ == page && (a+i)->padding\_ == padding)

found = false;

i++;

}

std::cout << a+i << "\nВведите что-нибудь, чтобы продолжить\n";

std::cin >> go\_next;

}

void do\_smth(data \*a, int prikol){

int i = 0 + rand() % (prikol-1) ;

(a+i)->appeal = (a+i)->appeal + (0 + rand() % 10);

}

void showTable(data \*a, int prikol){

for(int i = 0; i < 80;i++)

std::cout << "\_";

std::cout << "\n";

std::cout << "Имена процессов |" << " Номер страницы |" << " Сдвиг |\n";

for(int i = 0;i < 80;i++)

std::cout << "-";

std::cout << "\n";

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

std::cout << (a+i)->name;

makeSpaces(20 - (a+i)->name.length());

std::cout << "|" << (a+i)->page\_;

makeSpaces(27 - numLength((a+i)->page\_));

std::cout << "|" << (a+i)->padding\_;

makeSpaces(30 - numLength((a+i)->padding\_));

std::cout << "|\n";

}

for(int i = 0;i < 80;i++)

std::cout << "-";

std::cout << "\nВведите что-нибудь, чтобы продолжить\n";

std::string go;

std::cin >> go;

}

void remove(data \*a, int \*n, int page\_to\_kill, int padding\_to\_kill, float \*available\_memory, float size\_of\_page, int \*num\_of\_free\_pages){

bool i\_find\_it = false;

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

if((a+i)->page\_ == page\_to\_kill && (a+i)->padding\_ == padding\_to\_kill){

i\_find\_it = true;

\*(available\_memory) += size\_of\_page;

}

if(i\_find\_it)

\*(a+i) = \*(a+i+1);

}

if(i\_find\_it){

\*n = \*n - 1;

\*num\_of\_free\_pages = \*num\_of\_free\_pages + 1;

}

}

void addPlace(data \*a, int \*n, int \*j, int \*prikol, float \*available\_memory, int \*num\_of\_free\_pages, float size\_of\_page){

std::string name;

if(\*num\_of\_free\_pages == 0){

int page\_of\_min;

int padding\_of\_min;

int s;

s = (a+1)->appeal;

for(int i = 0;i < \*prikol;i++)

if((a+i)->appeal < s){

s = (a+i)->appeal;

page\_of\_min = (a+i)->page\_;

padding\_of\_min = (a+i)->padding\_;

}

remove(a, prikol, page\_of\_min, padding\_of\_min, available\_memory, size\_of\_page, num\_of\_free\_pages);

}

std::cout << "Введите имя процесса\n";

std::cin >> name;

\*num\_of\_free\_pages = \*num\_of\_free\_pages - 1;

\*(available\_memory) -= size\_of\_page;

(a+\*prikol)->name = name;

(a+\*prikol)->appeal = 0 + rand() % 10;

if(\*n > 10){

\*j = \*j + 1;

\*n = 0;

}

(a+\*prikol)->page\_ = \*j;

(a+\*prikol)->padding\_ = \*n;

\*n = \*n + 1;

\*prikol = \*prikol + 1;

}

int main(){

setlocale(LC\_ALL, "Russian");

bool dialog = true;

int your\_choose;

data memory[100];

int n = 0;

int j = 0;

int prikol = 0; // Кол-во элементов

float available\_memory;

int num\_of\_pages;

int num\_of\_free\_pages;

std::cout << "Введите кол-во доступной памяти(КБ)\n";

std::cin >> available\_memory;

std::cout << "Ввведите нужное кол-во страниц\n";

std::cin >> num\_of\_pages;

num\_of\_free\_pages = num\_of\_pages;

float size\_of\_page = (float)available\_memory / num\_of\_pages;

while(dialog){

system("cls");

giveInfo(available\_memory, num\_of\_pages, num\_of\_free\_pages, size\_of\_page);

std::cout << "Что вы хотите сделать?\n";

std::cin >> your\_choose;

switch(your\_choose){

case 1:

addPlace(&memory[0], &n, &j, &prikol, &available\_memory, &num\_of\_free\_pages, size\_of\_page);

break;

case 2:

int page\_to\_kill;

int padding\_to\_kill;

std::cout << "Введите номер страницы и отступ\n";

std::cin >> page\_to\_kill >> padding\_to\_kill;

remove(&memory[0], &prikol, page\_to\_kill, padding\_to\_kill, &available\_memory, size\_of\_page, &num\_of\_free\_pages);

break;

case 3:

showTable(&memory[0], prikol);

break;

case 4:

std::cout << "Введите номер страницы и отступ\n";

int page;

int padding;

std::cin >> page >> padding;

showAddress(page, padding, &memory[0], prikol);

break;

case 5:

do\_smth(&memory[0], prikol);

break;

case 6:

dialog = false;

break;

}

}

std::cout << "\nХорошего дня!";

return 0;

}