**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»

на тему: «Управление виртуальной памятью»

Выполнил: студент гр. ИТИ-11

Филончик К.В.

Принял: преподаватель-стажер Карась О.В.

Гомель 2022

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

**Задание:**

Менеджер памяти должен:

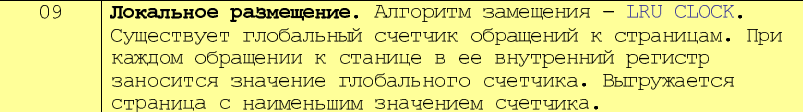
1. Разбивать память заданного размера на указанное количество страниц. На экран должна выводиться следующая информация о состоянии памяти: объем памяти, число страниц, число свободных страниц (%), размер страницы;

2. 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;

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

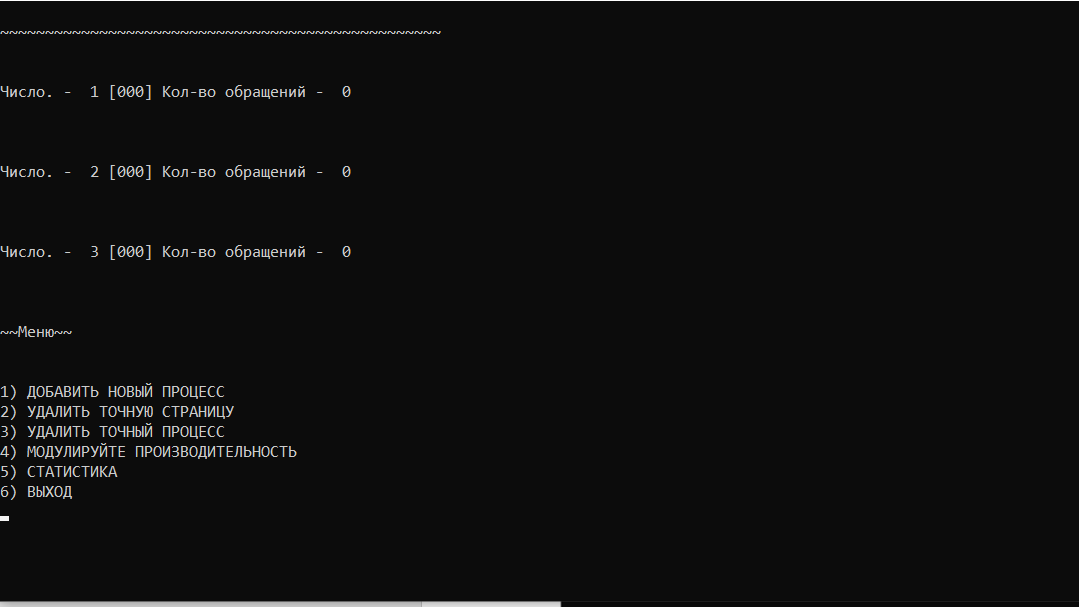
4. Организовывать циклическое обращение к страницам размещенным в памяти по нажатию на кнопку. При этом случайным образом задается количество обращений к страницам (диапазон 1..10). Для каждого обращения генерируется, случайным образом, номер страницы из диапазона [0; количество страниц памяти]. При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

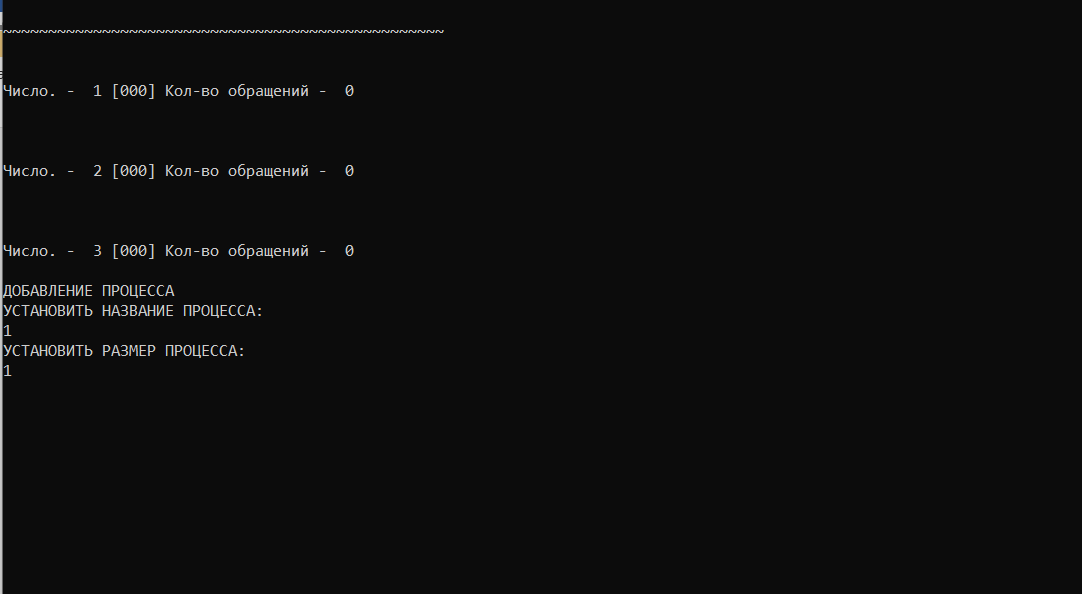
**Вариант 9**

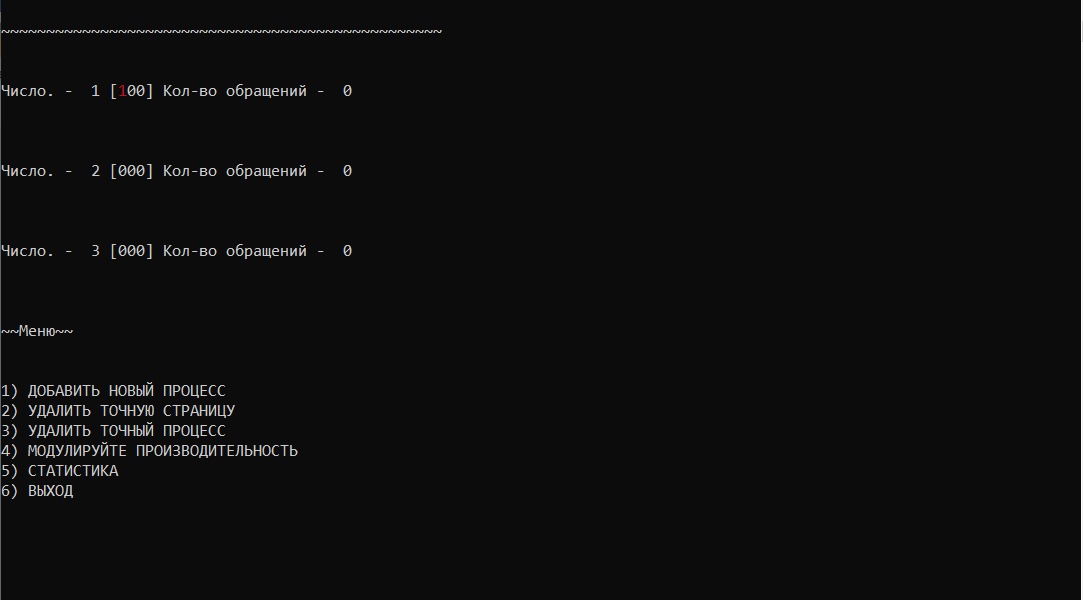
****

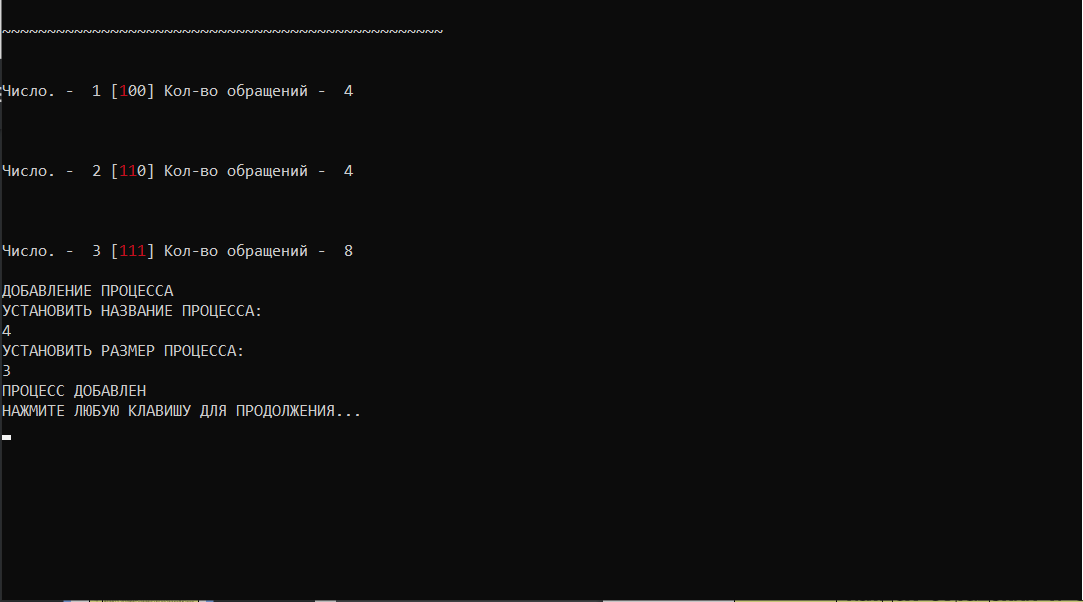
**Демонстрация функций:**

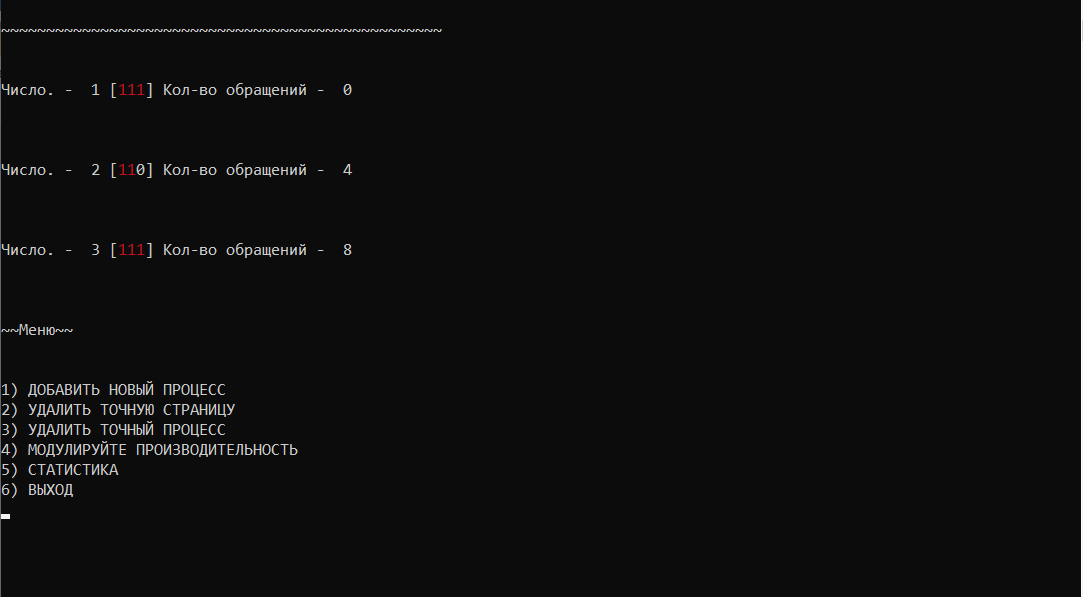


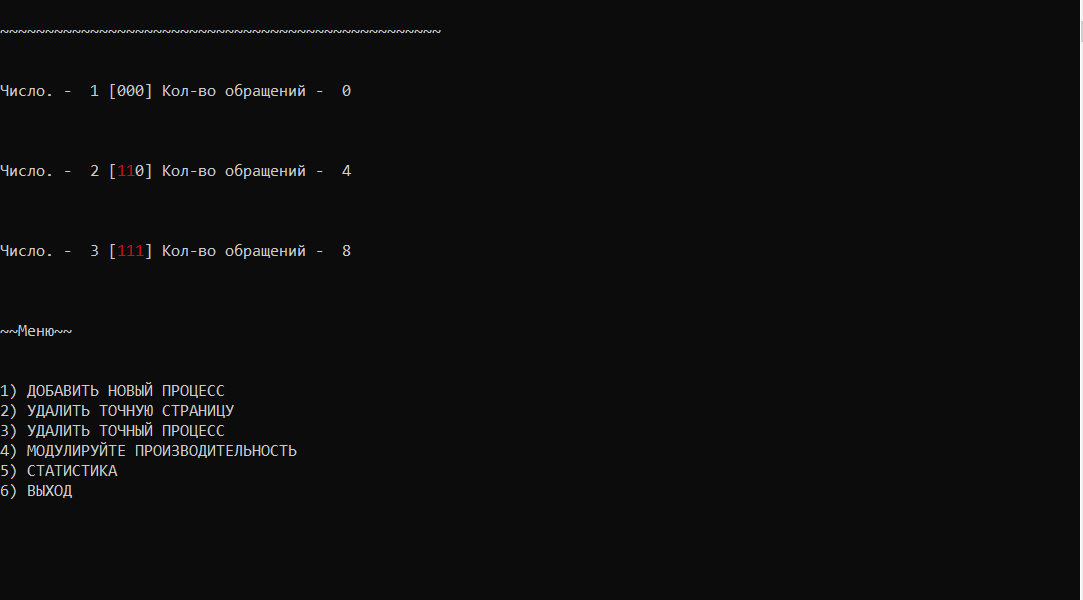


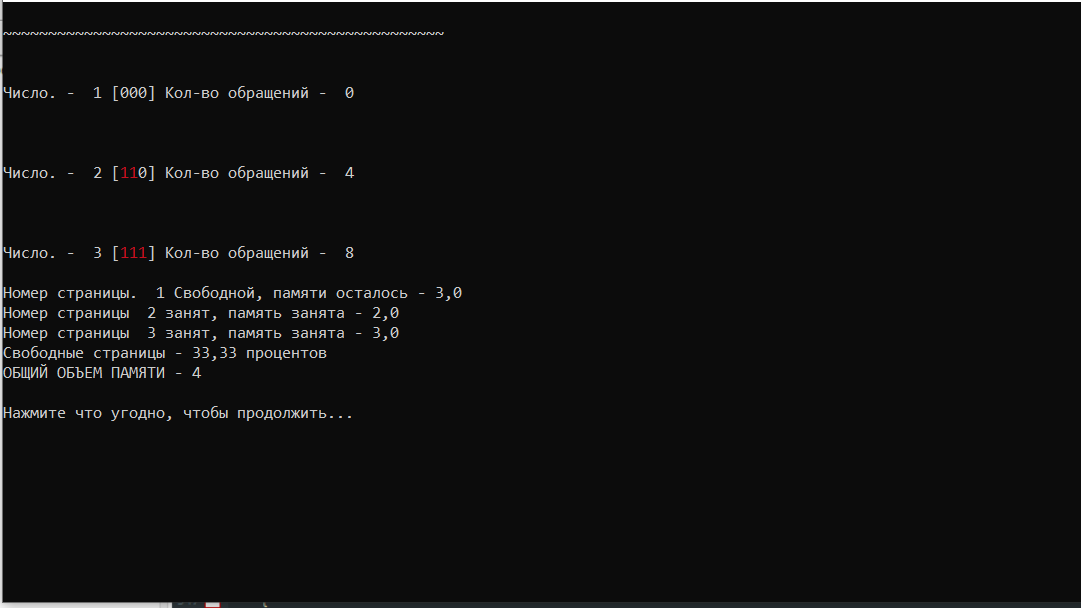












**Вывод:** в результате выполнения работы была разработана программа, реализующая заданный алгоритм замещения страниц в памяти.

**Листинг программы**

#include <stdio.h>

#include <stdlib.h>

#include <conio.h>

#include <windows.h>

#include <math.h>

#include <iostream>

HANDLE console = GetStdHandle(STD\_OUTPUT\_HANDLE);

struct pages

{

int memory[200];

int count;

};

struct processes

{

char name[20];

int size;

};

void memoryOut(struct pages \*page,int qP,int sP);

void workMod(struct pages \*page,int qP);

void addProc(struct pages \*page,int qP,int sP, struct processes \*process,int \*qProc,int \*freepages);

void delPage(struct pages \*page,int sP,int \*freepages);

void delProc(struct pages \*page,int qP,int sP, struct processes \*process,int \*qProc,int \*freepages);

void stats(struct pages \*page,int qP,int sP, struct processes \*process,int \*qProc,int \*freepages, int sizeM);

int main()

{

setlocale(LC\_ALL, "rus");

////////////////MEMORY PAGES

int sizeM,quanP,sizeP;//memory size, quantity of pages, size of pagede

puts("УСТАНОВИТЬ ОБЪЕМ ПАМЯТИ:");

scanf("%i",&sizeM);

puts("УСТАНОВИТЬ КОЛИЧЕСТВО СТРАНИЦ:");

scanf("%i",&quanP);

sizeP=sizeM/quanP;

struct pages page[quanP];

int i,j;

int freepages[quanP];

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

{

for(j=0;j<sizeP;j++)

{

page[i].memory[j]=0;

}

page[i].count=0;

freepages[i]=0;

}

////////////////PROCESSES

int qProc=0;//quantity of processes

struct processes process[100];

////////////////MENU

int item;//menu item

do

{

memoryOut(page,quanP,sizeP);

puts("\n");

puts("~~Меню~~");

puts("\n");

puts("1) ДОБАВИТЬ НОВЫЙ ПРОЦЕСС");

puts("2) УДАЛИТЬ ТОЧНУЮ СТРАНИЦУ");

puts("3) УДАЛИТЬ ТОЧНЫЙ ПРОЦЕСС");

puts("4) МОДУЛИРУЙТЕ ПРОИЗВОДИТЕЛЬНОСТЬ");

puts("5) СТАТИСТИКА");

puts("6) ВЫХОД");

scanf("%i",&item);

memoryOut(page,quanP,sizeP);

switch(item)

{

case 1:

{

addProc(page,quanP,sizeP,process,&qProc,freepages);

break;

}

case 2:

{

delPage(page,sizeP,freepages);

break;

}

case 3:

{

delProc(page,quanP,sizeP,process,&qProc,freepages);

break;

}

case 4:

{

workMod(page,quanP);

break;

}

case 5:

{

stats(page,quanP,sizeP,process,&qProc,freepages,sizeM);

puts("\nНажмите что угодно, чтобы продолжить...");

getch();

break;

}

case 6: {break;}

default:

{

puts("НЕПРАВИЛЬНЫЙ РЕГИСТР, пожалуйста, вставьте другой...");

getch();

break;

}

}

}

while(item!=6);

return 0;

}

void stats(struct pages \*page,int qP,int sP, struct processes \*process,int \*qProc,int \*freepages, int sizeM)

{

int i,j,k,local\_count=0;

float free=0, busy=0;

int free1=0, busy1=0;

float fp=0, bp=0;

float perc;

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

{

for(j=0;j<sP;j++)

{

if((page+i)->memory[j]!=0)

{

local\_count++;

busy++;

busy1++;

}

else

{

free++;

free1++;

}

}

if(local\_count==0)

{

if(sP==0)

{

printf("Номер страницы. %2i Свободной , памяти осталось - %.1f\n",i+1,fabs(sP-free));

}

else

{

printf("Номер страницы. %2i Свободной, памяти осталось - %.1f\n",i+1,free);

}

fp++;

}

else

{

if(sP==0)

{

printf("Номер страницы %2i занят, память занята - %.1f\n",i+1,fabs(sP-busy));

}

else

{

printf("Номер страницы %2i занят, память занята - %.1f\n",i+1,busy);

}

bp++;

}

local\_count=0;

free=0;

busy=0;

}

//printf("---%.0f-%.0f---", fp, bp);

perc=fp/(fp+bp);

printf("Свободные страницы - %.2f процентов", perc\*100);

printf("\nОБЩИЙ ОБЪЕМ ПАМЯТИ - %d\n",qP\*sP-busy1);

}

void memoryOut(struct pages \*page,int qP,int sP)

{

system("cls");

int i,j,k;

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

{

if(i%4==0)

{

printf("\n");

for(k=0;k<sP\*4+37;k++)

printf("~");

printf("\n");

}

else

puts("\n");

puts("\n");

printf("Число. - %2i [",i+1);

for(j=0;j<sP;j++)

{

if((page+i)->memory[j]==0)

printf("0");

else

{

SetConsoleTextAttribute(console,4);

printf("1");

SetConsoleTextAttribute(console,7);

}

}

printf("] Кол-во обращений - %2i",(page+i)->count);

}

printf("\n\n");

}

void workMod(struct pages \*page,int qP)

{

int ranNum=rand()%10+1;//random quantity of treatments

int ranP;//randomPage

int i;

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

{

ranP=rand()%qP;

(page+ranP)->count+=2;

}

}

void addProc(struct pages \*page,int qP,int sP, struct processes \*process,int \*qProc,int \*freepages)

{

setlocale(LC\_ALL, "rus");

int i,j,n=0;

puts("ДОБАВЛЕНИЕ ПРОЦЕССА");

puts("УСТАНОВИТЬ НАЗВАНИЕ ПРОЦЕССА:");

scanf("%s",(process+\*qProc)->name);

puts("УСТАНОВИТЬ РАЗМЕР ПРОЦЕССА:");

scanf("%i",&((process+\*qProc)->size));

int size=(process+\*qProc)->size;

int a=size/sP;//quantity of pages

int b=size%sP;//quantity of remaining memory

int freePage=0;

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

if(\*(freepages+i)==0)

freePage++;

if(b==0)

if(freePage>=a)

n=1;

if(b!=0)

if(freePage>a)

n=1;

if(n==1)

{

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

if(\*(freepages+i)==0 && size)

{

\*(freepages+i)=\*qProc+1;

for(j=0;j<sP && size;j++)

{

(page+i)->memory[j]=\*qProc+1;

size-=1;

}

freePage-=1;

}

}

else

{

if(size>qP\*sP)

{

puts("ОШИБКА: НЕДОСТАТОЧНО ОСТАЛОСЬ ПАМЯТИ");

puts("НАЖМИТЕ ЛЮБУЮ КЛАВИШУ ДЛЯ ПРОДОЛЖЕНИЯ...");

getch();

}

else

{

int minCount,flag,minPos;

while(size)

{

minCount=100;

minPos=qP-1;

for(i=qP-1;i>=0;i--)

{

if((page+i)->count<=minCount)

{

if(\*(freepages+i)!=\*qProc+1)

{

minCount=(page+i)->count;

minPos=i;

}

}

}

(page+minPos)->count+=2;

if(freePage)

{

flag=1;

for(i=0;i<qP && flag;i++)

if(\*(freepages+i)==0)

{

\*(freepages+i)=\*qProc+1;

flag=0;

freePage-=1;

for(j=0;j<sP && size;j++)

{

(page+i)->memory[j]=\*qProc+1;

(page+i)->count=0;

size-=1;

}

}

}

else if(size)

{

(page+minPos)->count=0;

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

{

(page+minPos)->memory[i]=0;

}

for(i=0;i<sP && size;i++)

{

(page+minPos)->memory[i]=\*qProc+1;

size-=1;

}

\*(freepages+minPos)=\*qProc+1;

}

(page+minPos)->count=0;

}

}

}

\*qProc+=1;

puts("ПРОЦЕСС ДОБАВЛЕН");

puts("НАЖМИТЕ ЛЮБУЮ КЛАВИШУ ДЛЯ ПРОДОЛЖЕНИЯ...");

getch();

}

void delPage(struct pages \*page,int sP,int \*freepages)

{

setlocale(LC\_ALL, "rus");

puts("УДАЛЕНИЕ СТРАНИЦЫ");

int number,i;

puts("ИНДЕКС СТРАНИЦЫ ДЛЯ УДАЛЕНИЯ:");

scanf("%i",&number);

number-=1;

(page+number)->count=0;

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

{

(page+number)->memory[i]=0;

}

\*(freepages+number)=0;

puts("СТРАНИЦА БЫЛА УДАЛЕНА");

puts("НАЖМИТЕ ЛЮБУЮ КЛАВИШУ ДЛЯ ПРОДОЛЖЕНИЯ...");

getch();

}

void delProc(struct pages \*page,int qP,int sP, struct processes \*process,int \*qProc,int \*freepages)

{

setlocale(LC\_ALL, "rus");

puts("УДАЛЕНИЕ ПРОЦЕССА");

int number,i,j;

puts("УКАЗАТЕЛЬ ПРОЦЕССА УДАЛЕНИЯ:");

scanf("%i",&number);

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

{

if(\*(freepages+i)==number)

{

\*(freepages+i)=0;

for(j=0;j<sP;j++)

{

(page+i)->memory[j]=0;

}

}

}

for(i=number-1;i<\*qProc-1;i++)

{

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

}

\*qProc-=1;

puts("ПРОЦЕСС УДАЛЕН");

puts("НАЖМИТЕ ЛЮБУЮ КЛАВИШУ ДЛЯ ПРОДОЛЖЕНИЯ...");

getch();

}