**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»**

Выполнила студентка гр. ИТИ-11:  
*Буякевич Д.О.*  
 Принял преподаватель:  
*Карась О.В.*

Гомель 2022

**Цель работы:** изучение алгоритмов управления памятью, разработка программы менеджера памяти.

**Задание**

Разработать программу, реализующую заданный алгоритм выделения памяти.

Менеджер памяти должен:

1. По запросу процесса выделять память, согласно заданного алгоритма (таблица 1). На экран должна выводиться следующая информация о состоянии памяти: объем памяти, объем свободной памяти, размер наибольшего свободного блока,

количество запросов на выделение памяти, количество удовлетворѐнных запросов (%).

2. Для выделения памяти указывается имя процесса и размер блока. После нажатия на кнопку «ДОБАВИТЬ» память выделяется или выдаѐтся сообщение о невозможности выделения.

3. Удалять из памяти заданный блок или все блоки заданного процесса (по нажатию кнопки «УДАЛИТЬ»). Указывается номер удаляемого блока и имя процесса.

4. Реализовать возможность последовательной записи/чтения информации в/из

выделенную память по логическому адресу. Вывести физического адреса ячейки памяти, в которую была осуществлена запись.

5. Организовывать циклическое выделение и освобождение памяти. При этом

случайным образом задается количество выделяемых блоков и их размер.



Рисунок 1 – Вариант лабораторной работы

**Ход работы:**

На рисунке 2 показано меню ЛБ и ввод кол-ва страниц, и размера памяти.

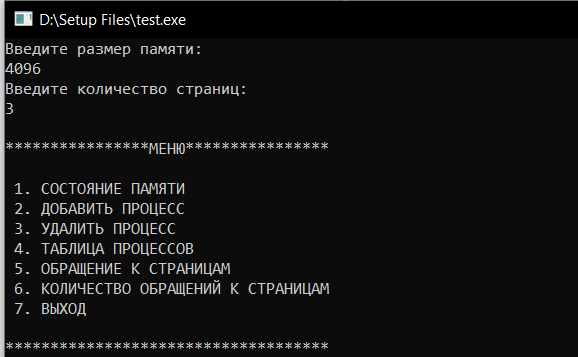


Рисунок 2 – Меню

На рисунке 3 показано состояние памяти

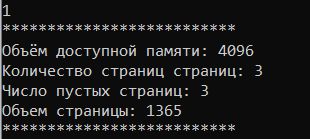


Рисунок 3 – Состояние памяти

На рисунке 4 показано добавление страниц

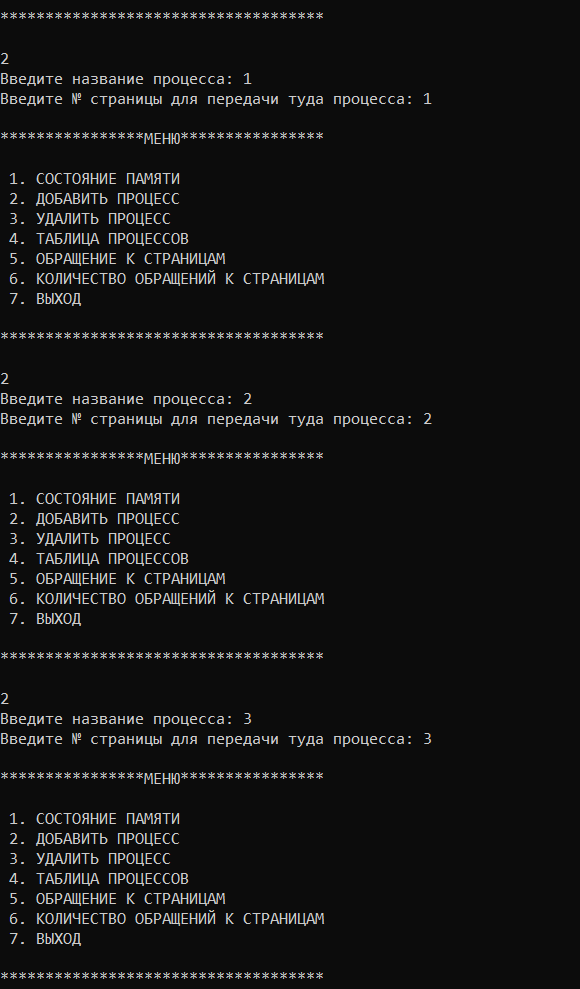


Рисунок 4 – Добавление процессов

На рисунке 5 – показано удаление страницы по номеру

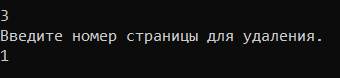


Рисунок 5 – Удаление страницы по номеру

На рисунке 6 – показаны оставшиеся процессы

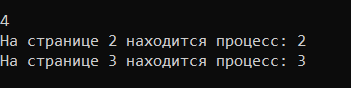


Рисунок 6 – Таблица процессов

На рисунке 7, выполнилось обращение к страницам и вывод кол-ва обращений.

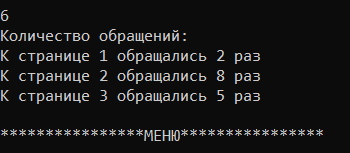


Рисунок 7 – Количество обращений

На рисунке 8 выход из консоли

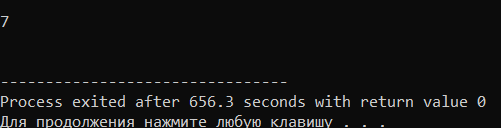


Рисунок 8 – Выход

**Вывод:** изучили алгоритмы замещения и размещения файлов на страницах виртуальной памяти.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

#include <stdio.h>

#include <string.h>

#include <locale.h>

#include <stdlib.h>

int avaibleProcesses=0;

int CanUsePages=0;

struct processes

{

char name[30];

int page;

int exist;

};

struct computer

{

int memory;

processes p[50];

int HowMuchPage;

int memoryToPage;

int \*PustoOrNoPage;

int \*deleteProcess;

int HowMuchDeleteProcesses;

int \*HowMuchObrasheniy;

};

computer computer;

processes process[50];

void showTable()

{

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

printf("Объём доступной памяти: %d\n",computer.memory);

printf("Количество страниц страниц: %d\n",computer.HowMuchPage);

printf("Число пустых страниц: %d\n",CanUsePages);

printf("Объем страницы: %d\n",computer.memoryToPage);

printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

}

void NewProcess()

{

int page,d=1,a=1,q=1,index1,index2;

char tmp[30];

fflush(stdin);

printf("Введите название процесса: ");

gets(process[avaibleProcesses].name);

printf("Введите № страницы для передачи туда процесса: ");

scanf("%d",&page);

while(a)

{

if (CanUsePages==0)

{

int randPage=rand()%computer.HowMuchPage;

while (page==randPage)

{

randPage=rand()%computer.HowMuchPage;

}

for(int i=0;i<avaibleProcesses && d;i++)

{

if(process[i].page==randPage)

{

index1=i;

d=0;

}

}

d=1;

for(int i=0;i<avaibleProcesses && d;i++)

{

if(process[i].page==page)

{

index2=i;

d=0;

}

}

printf("Процесс %s выгружен. На его место загружен процесс %s\n",process[index1].name,process[index2].name);

process[index2].page=process[index1].page;

strcpy(process[index2].name,process[index1].name);

process[index2].exist=0;

printf("Процесс %s загружен в страницу %d",process[avaibleProcesses].name,page);

process[avaibleProcesses].page=page;

process[avaibleProcesses].exist=1;

avaibleProcesses++;

a=0;

}

else

{

if (computer.PustoOrNoPage[page]==0)

{

computer.PustoOrNoPage[page]=1; // означает, что страница page занята процессом

process[avaibleProcesses].page=page;

process[avaibleProcesses].exist=1;

avaibleProcesses++;

a=0;

CanUsePages--;

}

else

{

puts("Данная страница не доступна на данный момент");

a=0;

}

}

}

}

void deleteProcess()

{

int number,index\_del,a=1;

puts("Введите номер страницы для удаления.");

scanf("%d",&number);

for(int i=0;i<avaibleProcesses && a;i++)

{

if(process[i].page==number)

{

index\_del=i;

a=0;

}

}

// computer.dlPage[computer.cdlPage]=number;

// computer.deleteProcess[computer.HowMuchDeleteProcesses]=index\_del;

// computer.cdlPage++;

// computer.HowMuchDeleteProcesses++;

process[index\_del].exist=0;

CanUsePages++;

computer.PustoOrNoPage[number]=0;

}

void showProcess()

{

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

{

if (process[i].exist)

{

printf("На странице %d находится процесс: %s \n",process[i].page,process[i].name);

}

}

}

void refferringToPages()

{

for (int i=0;i<computer.HowMuchPage;i++)

{

computer.HowMuchObrasheniy[i]+=rand()%10+1;

}

}

void showRefferences()

{

puts("Количество обращений: ");

for (int i=0;i<computer.HowMuchPage;i++)

{

printf("К странице %d обращались %d раз\n",i+1,computer.HowMuchObrasheniy[i]);

}

}

int main()

{

setlocale(LC\_ALL,"RUS");

puts("Введите размер памяти: ");

scanf("%d",&computer.memory);

puts("Введите количество страниц: ");

scanf("%d",&computer.HowMuchPage);

computer.memoryToPage=computer.memory/computer.HowMuchPage;

CanUsePages=computer.HowMuchPage;

computer.PustoOrNoPage=new int [computer.HowMuchPage+1];

computer.PustoOrNoPage=new int [computer.HowMuchPage+1];

computer.deleteProcess=new int [computer.HowMuchPage+1];

computer.HowMuchObrasheniy=new int[computer.HowMuchPage+1];

for(int j = 0; j < computer.HowMuchPage + 1; j++)

{

computer.PustoOrNoPage[j] = 0;

computer.HowMuchObrasheniy[j]=0;

}

int a=1,menu;

while (a)

{

puts("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*МЕНЮ\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

puts(" 1. СОСТОЯНИЕ ПАМЯТИ");

puts(" 2. ДОБАВИТЬ ПРОЦЕСС");

puts(" 3. УДАЛИТЬ ПРОЦЕСС");

puts(" 4. ТАБЛИЦА ПРОЦЕССОВ");

puts(" 5. ОБРАЩЕНИЕ К СТРАНИЦАМ");

puts(" 6. КОЛИЧЕСТВО ОБРАЩЕНИЙ К СТРАНИЦАМ");

puts(" 7. ВЫХОД\n");

puts("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

scanf("%d",&menu);

switch(menu)

{

case 1:

showTable();

break;

case 2:

NewProcess();

break;

case 3:

deleteProcess();

break;

case 4:

showProcess();

break;

case 5:

refferringToPages();

break;

case 6:

showRefferences();

break;

case 7:

a=0;

break;

default:

printf("Неверно введённые данные\n");

}

}

fflush(stdin);

getchar();

return 0;

}