**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.

Алгоритмы замещения страниц

Выполнил студент

группы ИТИ-11

*Громыко И. В.*

Проверил преподаватель

*Карась О.В.*

Гомель 2022

**Цель работы:** изучение алгоритмов управления виртуальной памятью, алгоритмов замещения страниц.

**Задание.**

Разработать программу, реализующую заданный алгоритм замещения страниц в памяти.

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.

**Ход работы**

**Порядок выполнения задания.**

Таблица 1

|  |  |
| --- | --- |
| Вариант | Задание |
| 6 | **Локальное размещение**. Алгоритм замещения – Random.  Замещается случайная страница. |

**Ход работы**

**Порядок выполнения задания.**

На рисунке 1 показан запуск программы.

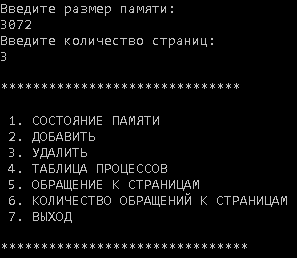


Рисунок 1 – Запуск программы

На рисунках 2 и 3 показано выполнения пункта 2 меню.

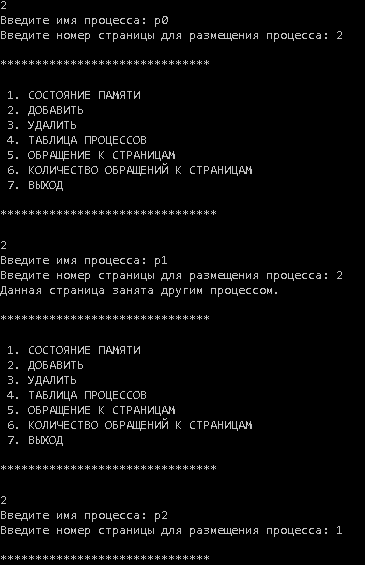


Рисунок 2 – Выполнение 2-го пункта меню

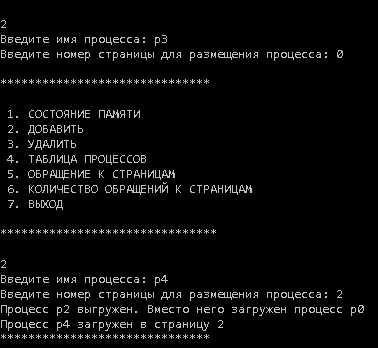


Рисунок 3 – Выполнение 2-го пункта меню

На рисунке 4 показано выполнения пункта 1 меню.

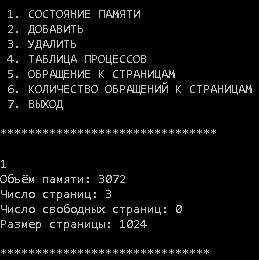


Рисунок 4 – Выполнение 1-го пункта меню

На рисунке 5 показано выполнения пункта 4 меню.

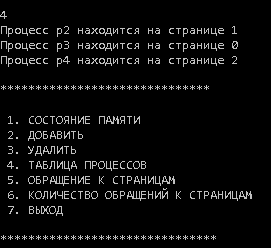


Рисунок 5 – Выполнение 4-го пункта меню

На рисунке 6 показано выполнения пункта 3 меню.

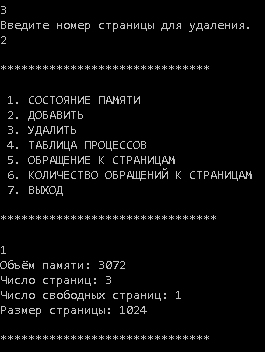


Рисунок 6 – Выполнение 3-го пункта меню

На рисунке 7 показано выполнения пункта 5 и 6 меню.

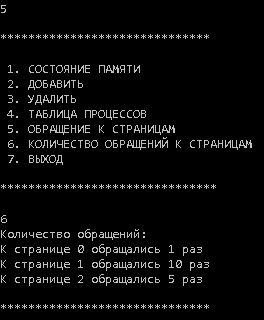


Рисунок 7 – Выполнение 5-го и 6-го пункта меню

На рисунке 8 показано выполнения пункта 7 меню.

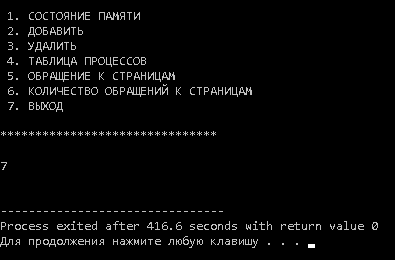


Рисунок 8 – Выполнение 7-го пункта меню

**Вывод:** в процессе работы изучили алгоритмы управления виртуальной памятью, разработали программу с определёнными алгоритмами замещения и размещения.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

**Листинг программы laba:**

#include <stdio.h>

#include <string.h>

#include <locale.h>

#include <stdlib.h>

int cprocesses=0;

int freePages=0;

struct processes

{

char name[30];

int page;

int exist;

};

struct computer

{

int memory;

processes p[50];

int numberPages;

int pageMemory;

int \*frPage;

int \*dlPage;

int cdlPage; // кол-во удалённых страниц

int \*dlProcess;

int cdlProcess; //кол-во удалённых прцоессов

int \*rfPages; //массив для кол-ва обращений

};

computer comp;

processes process[50];

void showData()

{

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

printf("Число страниц: %d\n",comp.numberPages);

printf("Число свободных страниц: %d\n",freePages);

printf("Размер страницы: %d\n",comp.pageMemory);

}

void addProcess()

{

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

char tmp[30];

fflush(stdin);

printf("Введите имя процесса: ");

gets(process[cprocesses].name);

printf("Введите номер страницы для размещения процесса: ");

scanf("%d",&page);

while(t)

{

if (freePages==0)

{

int randPage=rand()%comp.numberPages;

while (page==randPage)

{

randPage=rand()%comp.numberPages;

}

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

{

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

{

index1=i;

d=0;

}

}

d=1;

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

{

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

{

index2=i;

d=0;

}

}

printf("Процесс %s выгружен. Вместо него 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[cprocesses].name,page);

process[cprocesses].page=page;

process[cprocesses].exist=1;

cprocesses++;

t=0;

}

else

{

if (comp.frPage[page]==0)

{

comp.frPage[page]=1;

process[cprocesses].page=page;

process[cprocesses].exist=1;

cprocesses++;

t=0;

freePages--;

}

else

{

puts("Данная страница занята другим процессом.");

t=0;

}

}

}

}

void deleteProcess()

{

int number,index,t=1;

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

scanf("%d",&number);

for(int i=0;i<cprocesses && t;i++)

{

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

{

index=i;

t=0;

}

}

comp.dlPage[comp.cdlPage]=number;

comp.dlProcess[comp.cdlProcess]=index;

comp.cdlPage++;

comp.cdlProcess++;

process[index].exist=0;

freePages++;

comp.frPage[number]=0;

}

void showProcess()

{

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

{

if (process[i].exist)

{

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

}

}

}

void refferringToPages()

{

for (int i=0;i<comp.numberPages;i++)

{

comp.rfPages[i]+=rand()%10+1;

}

}

void showRefferences()

{

puts("Количество обращений: ");

for (int i=0;i<comp.numberPages;i++)

{

printf("К странице %d обращались %d раз\n",i,comp.rfPages[i]);

}

}

int main()

{

setlocale(LC\_ALL,"RUS");

puts("Введите размер памяти: ");

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

puts("Введите количество страниц: ");

scanf("%d",&comp.numberPages);

comp.pageMemory=comp.memory/comp.numberPages;

freePages=comp.numberPages;

comp.frPage=new int [comp.numberPages+1];

comp.dlPage=new int [comp.numberPages+1];

comp.dlProcess=new int [comp.numberPages+1];

comp.rfPages=new int[comp.numberPages+1];

for(int j = 0; j < comp.numberPages + 1; j++)

{

comp.frPage[j] = 0;

comp.rfPages[j]=0;

}

int t=1,menu;

while (t)

{

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

puts(" 1. СОСТОЯНИЕ ПАМЯТИ");

puts(" 2. ДОБАВИТЬ");

puts(" 3. УДАЛИТЬ");

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:

showData();

break;

case 2:

addProcess();

break;

case 3:

deleteProcess();

break;

case 4:

showProcess();

break;

case 5:

refferringToPages();

break;

case 6:

showRefferences();

break;

case 7:

t=0;

break;

default:

printf("Неверно введённые данные\n");

}

}

fflush(stdin);

getchar();

return 0;

}