**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.

Алгоритмы замещения страниц»

Выполнил: студент гр. ИТП-11

Иванов Д.А.

Принял: преподаватель

Карась О.В.

Гомель 2022

**Цель:** изучить основные алгоритмы замещение страниц и работой с виртуальной памятью.

**Задача:**

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

1. Разбивать память заданного размера на указанное количество страниц. На экран должна выводится 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 12:

Локальное размещение с динамическим увеличением количества выделенных страниц при количестве запросов к страницам больше установленного порога. Алгоритм замещения – FIFO. Реализуется очередь страниц в конец которой попадают страницы размещенные в памяти, а из начала берутся для замещения.

**Задание**

*FIFO* — способ организации и манипулирования данными относительно времени и приоритетов. В соответствии с вариантом программа должна размещать новые процессы в конце страницы, а процессы, которые были первыми добавлены в память будут замещается со страницы. При запуске программы появляется меню, где пользователь может выбрать какое количество страниц, памяти и процессов будет использовать пользователь. Пример меню выбора страниц и памяти представлен на рисунке 1.



Рисунок 1 - меню выбора страниц и памяти

После этого на экран выводится меню, где показана вся информация, которую ввел пользователь, и меню, где пользователь может добавить страницу в память, запустить программу, удалить страницу или выйти из программы. Пример главного меню программы представлен на рисунке 2.

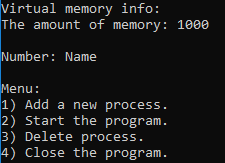


Рисунок 2 – главное меню программы

При выборе первого меню программы открывается окошко, где пользователь указывает номер процесса и его имя. Пример этого меню указан на рисунке 3.



Рисунок 3 – меню для добавления страницы в памяти

После добавления нескольких страниц мы можем в главном меню выбрать второй пункт и запустить программу. Результат на рисунках 4, 5.1, 5.2 и 6.1, 6.2, 6.3.

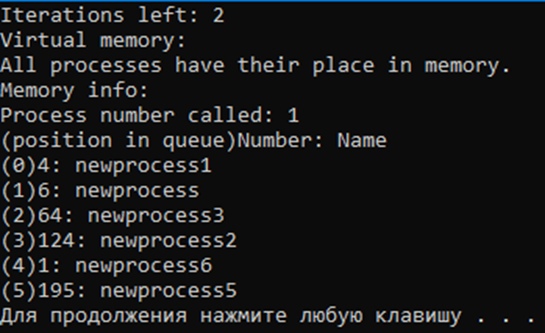


Рисунок 4 – результат работы программы (заполнение)

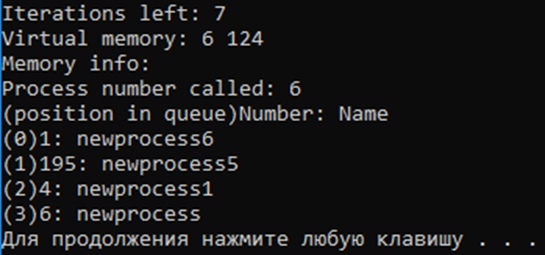


Рисунок 5.1 – результат работы программы (замещение)

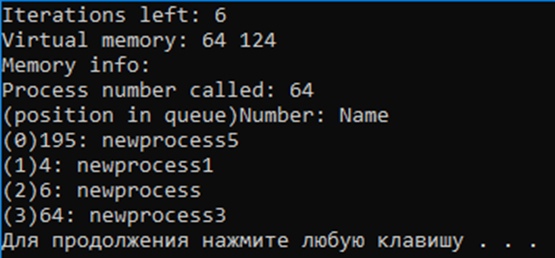


Рисунок 5.2 – результат работы программы (замещение)

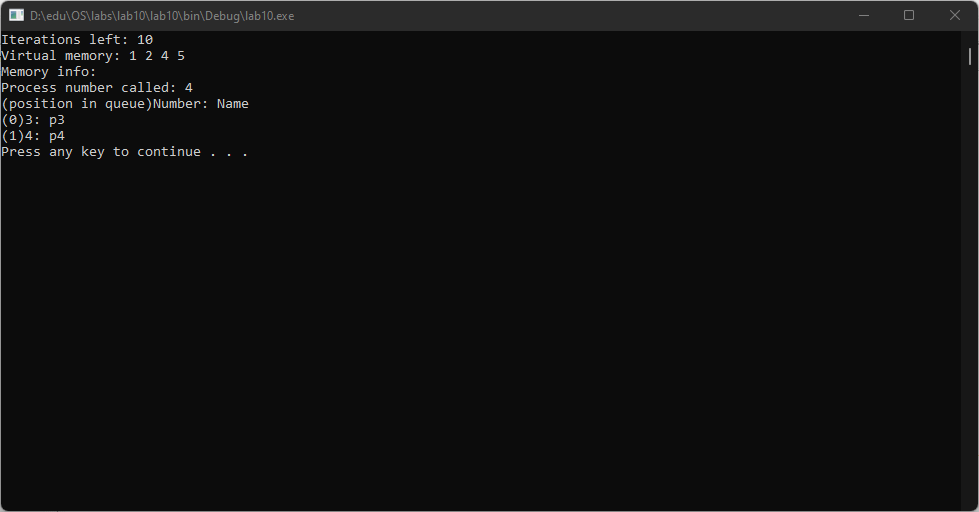


Рисунок 6.1 – результат работы программы (добавление страниц памяти)

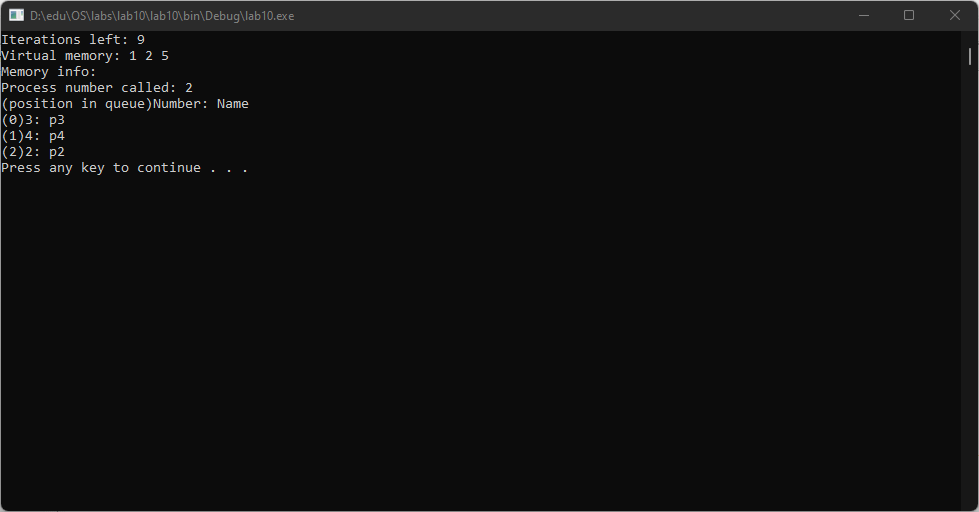


Рисунок 6.2 – результат работы программы (добавление страниц памяти)

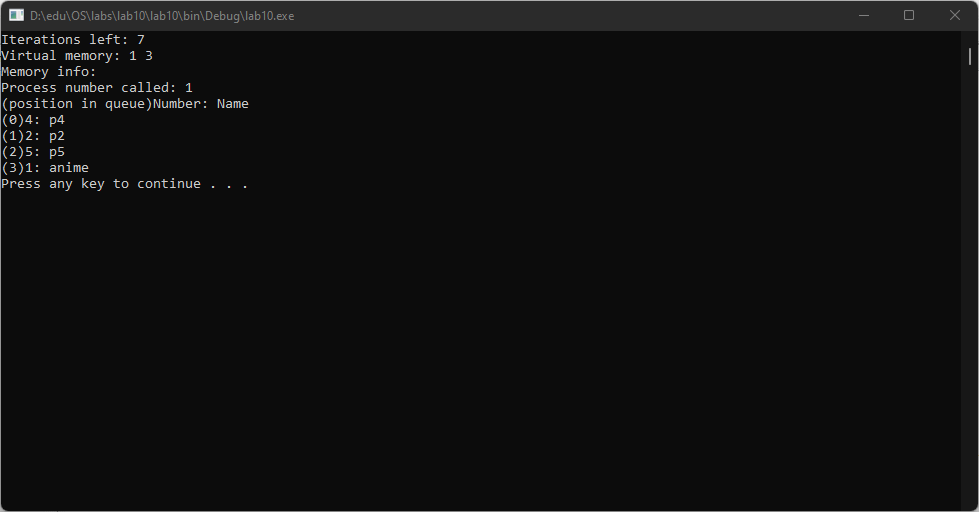


Рисунок 6.3 – результат работы программы (добавление страниц памяти)

**Вывод**: в ходе лабораторной работы были изучены основные алгоритмы замещение страниц и работа с виртуальной памятью.

**Листинг программы**

#include <bits/stdc++.h>

using namespace std;

map<int, pair<string, bool> > proc;

vector<int> mem;

int m, n, t, t\_n;

void proc\_call()

{

if (t == 0 && proc.size() > t\_n) {

t = 2;

t\_n++;

}

vector<int> free\_proc, any\_proc;

cout << "Virtual memory: ";

for (auto& i : proc)

if (!i.second.second) { free\_proc.push\_back(i.first); cout << i.first << ' '; }

else any\_proc.push\_back(i.first);

cout << endl;

if (free\_proc.size() == 0) {

cout << "All processes have their place in memory." << endl;

cout << "Memory info:" << endl << "Process number called: " << any\_proc[rand() % any\_proc.size()] << endl << "(position in queue)Number: Name" << endl;

for (int i = 0; i < mem.size(); i++) {

cout << "(" << i << ")" << mem[i] << ": " << proc[mem[i]].first << endl;

}

system("pause");

return;

}

int r\_proc = free\_proc[rand() % free\_proc.size()];

if (mem.size() == t\_n) {

proc[mem[0]].second = false;

mem.erase(mem.begin());

mem.push\_back(r\_proc);

proc[r\_proc].second = true;

}

else {

mem.push\_back(r\_proc);

proc[r\_proc].second = true;

}

cout << "Memory info:" << endl << "Process number called: " << r\_proc << endl << "(position in queue)Number: Name" << endl;

for (int i = 0; i < mem.size(); i++) {

cout << "(" << i << ")" << mem[i] << ": " << proc[mem[i]].first << endl;

}

t--;

system("pause");

}

int main()

{

cout << "Enter the amount of memory: ";

cin >> m;

cout << "Enter number of pages: ";

cin >> n;

if (m % n != 0) cout << m << " is not a multiple of " << n << ". So some memory will be wasted." << endl;

system("pause");

int p = 0;

while (p != 4) {

system("cls");

cout << "Virtual memory info:" << endl << "The amount of memory: " << m - (m % n) << endl << endl << "Number: Name" << endl;

for (auto& i : proc) {

cout << i.first << ": " << i.second.first << endl;

i.second.second = false;

}

cout << endl << "Menu:" << endl << "1) Add a new process." << endl << "2) Start the program." << endl << "3) Delete process." << endl << "4) Close the program." << endl;

cin >> p;

system("cls");

string temp\_n;

int temp\_m = 0;

switch (p) {

case 1:

{

cout << "Enter the name of the new process: ";

cin.ignore();

getline(cin, temp\_n);

cout << "Enter the number of this process: ";

cin >> temp\_m;

proc[temp\_m] = make\_pair(temp\_n, false);

break;

}

case 2:

{

cout << "Enter the request threshold at which the number of pages will increase(after the the memory is full): ";

cin >> t;

t += n;

t\_n = n;

int temp = rand() % 30;

mem.clear();

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

system("cls");

cout << "Iterations left: " << temp - i << endl;

proc\_call();

}

break;

}

case 3:

{

cout << "Enter the number of the process: ";

cin >> temp\_m;

auto it = proc.find(temp\_m);

if (it != proc.end()) {

proc.erase(it);

}

else {

cout << "There are no processes with that name." << endl;

system("pause");

}

break;

}

}

}

}