МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА №  43

ОТЧЁТ

ЗАЩИЩЁН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

### Старший преподаватель                                                      Поляк М.Д.

должность, уч. Степень, звание   подпись, дата                    инициалы, фамилия

ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №4.

Управление памятью.

по курсу: [Операционные системы](https://pro.guap.ru/inside/students/subjects/3154495)

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. 4136                                                                                Бобрович Н. С.

                                                                         подпись, дата                      инициалы, фамилия

Санкт-Петербург 2024

1. **Цель работы:**

Знакомство с принципами организации виртуальной памяти.

1. **Задание:**

В данной работе необходимо реализовать фрагмент диспетчера памяти и часть функционала операционной системы, отвечающего за замещение страниц при возникновении ошибок отсутствия страниц. Для упрощения работы предполагается использование линейной инвертированной таблицы страниц, работу с которой необходимо реализовать в виде программы. Также для простоты предполагается, что в системе имеется один единственный процесс, поэтому идентификатор процесса в инвертированной таблице страниц не хранится. Входные данные представляют собой последовательность операций обращения к памяти, выходные данные - состояние инвертированной таблицы страниц после каждой операции обращения к памяти.

Вычислить номер варианта по списку в журнале и сохранить его в файл [TASKID.txt](https://github.com/suai-os-2024/os-task4-OskolockKoli/blob/master/TASKID.txt) в репозитории.

Написать программу на языке C++ в соответствии со следующей спецификацией.

* 1. Входные данные:
     1. Аргумент командной строки (число): номер алгоритма замещения страниц, который должна использовать программа. Принимает значения 1 или 2, соответствующие двум алгоритмам замещения страниц, заданным по варианту.
     2. Перечень инструкций обращения к памяти, считываемый программой из стандартного потока ввода. На каждой строке не более одной инструкции. Инструкция состоит из двух чисел, разделенных пробелом, например: 0 1. Первое число обозначает тип операции доступа к памяти: 0 - чтение и 1 - запись. Второе число является номером виртуальной страницы, к которой происходит обращение.
  2. Выходные данные:
     1. Для каждой операции обращения к памяти, информация о которой поступила на вход программы, на выходе должна быть сгенерирована строка, содержащая содержимое инвертированной таблицы страниц в виде последовательности номеров виртуальных страниц, разделенных пробелом. Если какая-либо из записей в таблице страниц отсутствует (таблица страниц не заполнена до конца), вместо номера виртуальной страницы необходимо вывести символ #.

Весь код поместить в файле lab4.cpp. Код должен корректно компилироваться командой g++ lab4.cpp -o lab4 -std=c++11. Настоятельно рекомендуется использовать стандартную библиотеку STL. Полезными могут быть контейнеры [list](https://en.cppreference.com/w/cpp/container/list), [vector](https://en.cppreference.com/w/cpp/container/vector), [bitset](https://en.cppreference.com/w/cpp/utility/bitset) и др.

Если в работе алгоритма замещения страниц используется бит R, то необходимо реализовать эмуляцию прерывания таймера. Для этого через каждые 5 операций обращения к памяти необходимо запускать обработчик данного прерывания. Значения битов R по прерыванию таймера сбрасываются.

Для алгоритмов, использующих счетчик (NFU, Aging): если несколько страниц имеют одинаковое значение счетчика, одна из них выбирается случайным образом. При повторной загрузке страницы в память ее счетчик обнуляется. В алгоритме старения счетчик имеет размер 1 байт. В алгоритме NFU счетчик имеет размер не меньше 4 байт.

Во всех алгоритмах, использующих датчик случайных чисел (Random, NRU, NFU, Aging, ...), разрешается использовать **только** функцию int uniform\_rnd(int a, int b), объявленную в файле [lab4.h](https://github.com/suai-os-2024/os-task4-OskolockKoli/blob/master/lab4.h). Данная функция генерирует случайное целое число с равномерным распределением из диапазона [a, b]. Использование других функций для работы со случайными числами запрещено!

В качестве системного времени в алгоритме рабочего набора следует использовать количество инструкций доступа к памяти, обработанных с момента запуска программы.

После успешного прохождения локальных тестов необходимо загрузить код в репозиторий на гитхабе.

Сделать выводы об эффективности реализованных алгоритмов замещения страниц. Сравнить количество ошибок отсутствия страниц, генерируемых на тестовых данных при использовании каждого алгоритма.

Подготовить отчет о выполнении лабораторной работы и загрузить его под именем report.pdf в репозиторий. В случае использования системы компьютерной верстки LaTeX также загрузить исходный файл report.tex.

**Вариант 3:**

Скриншот 07-06-2024 140229Скриншот 07-06-2024 140218

1. **Описание используемых алгоритмов замещения страниц:**

**Clock:**

Дальнейшая модификация алгоритма Второй шанс. Поскольку перемещать страницы из головы очереди в конец весьма неэффективно, для увеличения скорости работы алгоритма Второй шанс очередь может быть реализована в виде циклического списка. В этом случае фактического перемещения страниц из головы в конец очереди не происходит. Вместо этого изменяется значение переменной, указывающей на голову очереди. В этом алгоритме переменная-указатель ("стрелка" часов) всегда указывает на голову списка, т.е. на страницу, загруженную раньше всех остальных.

**NRU:**

В соответствии со значениями битов использования и модификации (R и M), каждый страничный блок относят к одному из 4-х классов:

* класс 0, страница не использовалась и не была модифицирована;
* класс 1, страница не использовалась, но была модифифирована;
* класс 2, страница использовалась, но не была модифицирована;
* класс 3, страница использовалась и была модифицирована.

Удаляют страницу, относящуюся к классу с наименьшим номером. Если таких страниц несколько, одну из них выбирают случайным образом.

1. **Результат выполнения работы:**

Output1.txt:

10 # # # # # # #

10 30 # # # # # #

10 30 4 # # # # #

10 30 4 27 # # # #

10 30 4 27 15 # # #

10 30 4 27 15 19 # #

10 30 4 27 15 19 14 #

10 30 4 27 15 19 14 43

24 30 4 27 15 19 14 43

24 18 4 27 15 19 14 43

24 18 5 27 15 19 14 43

24 18 5 30 15 19 14 43

24 18 5 30 10 19 14 43

24 18 5 30 10 2 14 43

24 18 5 30 10 2 4 43

24 18 5 30 10 2 4 34

7 18 5 30 10 2 4 34

7 14 5 30 10 2 4 34

7 14 55 30 10 2 4 34

7 14 55 28 10 2 4 34

7 14 55 28 10 2 4 34

7 14 55 28 40 2 4 34

7 14 55 28 40 8 4 34

7 14 55 28 40 8 44 34

7 14 55 28 40 8 44 49

1 14 55 28 40 8 44 49

1 21 55 28 40 8 44 49

1 21 55 28 40 8 44 49

1 21 0 28 40 8 44 49

1 21 0 28 40 8 44 49

1 21 0 54 40 8 44 49

1 21 0 54 35 8 44 49

1 21 0 54 35 15 44 49

1 21 0 54 35 15 27 49

1 21 0 54 35 15 27 49

1 21 0 54 35 15 27 49

1 21 0 54 35 15 27 46

24 21 0 54 35 15 27 46

24 25 0 54 35 15 27 46

24 25 0 8 35 15 27 46

24 25 0 8 3 15 27 46

24 25 0 8 3 38 27 46

24 25 0 8 3 38 27 46

24 25 0 8 3 38 40 46

24 25 0 8 3 38 40 22

17 25 0 8 3 38 40 22

17 36 0 8 3 38 40 22

17 36 44 8 3 38 40 22

17 36 44 31 3 38 40 22

17 36 44 31 42 38 40 22

17 36 44 31 42 4 40 22

17 36 44 31 42 4 40 22

17 36 44 31 42 4 38 22

17 36 44 31 42 4 38 11

6 36 44 31 42 4 38 11

6 7 44 31 42 4 38 11

6 7 5 31 42 4 38 11

6 7 5 31 42 4 38 11

6 7 5 10 42 4 38 11

6 7 5 10 42 4 38 11

Output2.txt:

10 # # # # # # #

10 30 # # # # # #

10 30 4 # # # # #

10 30 4 27 # # # #

10 30 4 27 15 # # #

10 30 4 27 15 19 # #

10 30 4 27 15 19 14 #

10 30 4 27 15 19 14 43

10 30 4 24 15 19 14 43

10 30 4 24 18 19 14 43

10 30 4 24 18 5 14 43

10 30 4 24 18 5 14 43

10 30 4 24 18 5 14 43

10 30 4 24 2 5 14 43

10 30 4 24 2 5 14 43

10 30 4 24 34 5 14 43

10 30 4 24 34 7 14 43

10 30 4 24 34 7 14 43

10 30 4 55 34 7 14 43

10 30 28 55 34 7 14 43

10 30 28 55 34 7 14 43

10 30 28 55 34 40 14 43

10 30 28 8 34 40 14 43

10 30 28 8 44 40 14 43

10 30 28 8 44 40 49 43

10 30 28 8 44 40 1 43

10 30 28 21 44 40 1 43

10 30 8 21 44 40 1 43

10 30 8 21 44 0 1 43

10 30 8 21 44 0 1 43

10 30 8 21 44 0 54 43

10 30 8 21 44 35 54 43

10 30 8 15 44 35 54 43

10 30 27 15 44 35 54 43

21 30 27 15 44 35 54 43

21 30 27 15 44 35 0 43

21 30 46 15 44 35 0 43

21 30 46 24 44 35 0 43

25 30 46 24 44 35 0 43

25 8 46 24 44 35 0 43

3 8 46 24 44 35 0 43

3 38 46 24 44 35 0 43

3 38 46 24 44 35 8 43

3 38 40 24 44 35 8 43

3 38 40 24 44 35 8 22

3 38 40 24 44 35 8 17

3 38 36 24 44 35 8 17

3 38 36 24 44 35 8 17

31 38 36 24 44 35 8 17

31 42 36 24 44 35 8 17

4 42 36 24 44 35 8 17

4 42 36 24 44 35 8 17

4 42 38 24 44 35 8 17

4 11 38 24 44 35 8 17

4 11 38 24 44 35 6 17

4 11 38 24 44 35 7 17

4 11 5 24 44 35 7 17

6 11 5 24 44 35 7 17

6 10 5 24 44 35 7 17

6 10 5 42 44 35 7 17

1. **Исходный код программы с комментариями:**

#include <iostream>

#include <deque>

#include <sstream>

#include <string>

#include <vector>

#include "lab4.h"

using namespace std;

// Размер массива

static int const sizeArray = 8;

// Счетчик операций

static int counter = 0;

// Указатель на текущий элемент в clockArray

int clockPtr = 0;

// Структура для хранения информации о странице в памяти

struct ClockRow {

int index; // Номер страницы

int read; // Флаг, показывающий, была ли страница недавно использована

};

// Динамический массив для хранения информации о страницах в памяти

deque<ClockRow> clockArray;

// Функция для вывода содержимого clockArray

void outClock() {

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

if (i < clockArray.size()) {

cout << clockArray[i].index;

}

else {

cout << "#";

}

if (i < sizeArray - 1) {

cout << " ";

}

}

cout << "\n";

}

// Функция для проверки наличия страницы в clockArray

bool isNotExist(int pageNumber) {

for (auto& row : clockArray) {

if (row.index == pageNumber) {

row.read = 1;

return false;

}

}

return true;

}

// Основная функция, реализующая алгоритм замещения страниц на основе циферблата (Clock)

void alClock() {

string line;

int firstNumOper = 0;

int secNumRow = 0;

while (true) {

getline(cin, line);

if (line.empty()) {

return;

}

istringstream istream{ line };

if (!(istream >> firstNumOper) || !(istream >> secNumRow) || istream >> line) {

continue;

}

// Сбрасывать флаг "использовалась" для всех страниц каждые 5 операций

if (counter == 5)

{

for (auto& row : clockArray) {

row.read = 0;

}

counter = 0;

}

counter++;

// Если страница с указанным номером не существует в clockArray

if (isNotExist(secNumRow)) {

// Если размер clockArray равен размеру массива

if (clockArray.size() == sizeArray) {

// Найти страницу для замещения, используя алгоритм циферблата (Clock)

while (true) {

if (clockArray[clockPtr % sizeArray].read) {

clockArray[clockPtr % sizeArray].read = 0;

clockPtr++;

continue;

}

else {

clockArray[clockPtr % sizeArray].index = secNumRow;

clockArray[clockPtr % sizeArray].read = 1;

clockPtr++;

break;

}

}

}

else {

// Добавить новую страницу в clockArray

clockArray.push\_back({ secNumRow, 1 });

}

}

// Вывести текущее состояние clockArray

outClock();

}

}

// Структура для хранения информации о странице в памяти для алгоритма NRU

struct ElementNRU {

int index; // Номер страницы

bool read; // Флаг, показывающий, была ли страница недавно использована

bool modify; // Флаг, показывающий, была ли страница изменена

};

// Динамический массив для хранения информации о страницах в памяти для алгоритма NRU

vector<ElementNRU> nruArray;

// Функция для проверки наличия страницы в nruArray

bool nruisNotExist(int index) {

for (auto& row : nruArray) {

if (row.index == index) {

row.read = true;

return false;

}

}

return true;

}

int getIndex(vector<int> classVec) {

// Если вектор classVec не пустой

if (!classVec.empty()) {

// Если размер вектора больше 1

if (classVec.size() != 1) {

// Возвращаем случайный индекс из диапазона [0, classVec.size() - 1]

return classVec[uniform\_rnd(0, classVec.size() - 1)];

}

else {

// Возвращаем единственный элемент вектора

return classVec[0];

}

}

// Если вектор пустой, возвращаем -1

return -1;

}

void nruImpl(int firstNumOper, int secNumRow) {

// Сбрасывать флаг "использовалась" для всех страниц каждые 5 операций

if (counter == 5)

{

for (auto& row : nruArray) {

row.read = 0;

}

counter = 0;

}

counter++;

// Если страница с указанным номером не существует в nruArray

if (nruisNotExist(secNumRow)) {

// Отмечаем, была ли страница изменена

bool isModifyed = firstNumOper == 1;

// Если размер nruArray равен размеру массива

if (nruArray.size() == sizeArray) {

// Разделяем страницы в nruArray на 4 категории в соответствии с алгоритмом NRU

vector<int> rang0;

vector<int> rang1;

vector<int> rang2;

vector<int> rang3;

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

if (!nruArray[i].read && !nruArray[i].modify) {

rang0.push\_back(i);

}

else if (!nruArray[i].read && nruArray[i].modify) {

rang1.push\_back(i);

}

else if (nruArray[i].read && nruArray[i].modify) {

rang2.push\_back(i);

}

else if (nruArray[i].read && nruArray[i].modify) {

rang3.push\_back(i);

}

}

// Выбираем страницу для замещения, используя алгоритм NRU

int index = getIndex(rang0);

if (index == -1)

index = getIndex(rang1);

if (index == -1)

index = getIndex(rang2);

if (index == -1)

index = getIndex(rang3);

// Заменяем выбранную страницу на новую

nruArray[index].index = secNumRow;

nruArray[index].read = true;

nruArray[index].modify = isModifyed;

}

else

{

// Добавляем новую страницу в nruArray

nruArray.push\_back({ secNumRow, true, isModifyed });

}

}

}

void outNRU() {

// Выводим содержимое nruArray

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

if (i < nruArray.size()) {

cout << nruArray[i].index;

}

else {

cout << "#";

}

if (i < sizeArray - 1) {

cout << " ";

}

}

cout << "\n";

}

void aNru() {

// Читаем ввод из консоли

string inputString;

int firstNumOper = 0;

int secNumRow = 0;

while (true) {

getline(cin, inputString);

if (inputString.empty()) {

return;

}

istringstream istream{ inputString };

if (!(istream >> firstNumOper) || !(istream >> secNumRow) || istream >> inputString) {

continue;

}

// Вызываем функцию nruImpl для обработки полученного ввода

nruImpl(firstNumOper, secNumRow);

// Выводим текущее состояние nruArray

outNRU();

}

}

int main(int argc, char\* argv[]) {

// Получаем номер аргумента из командной строки

int argoNumber = atoi(argv[1]);

// Выбираем алгоритм замещения страниц в зависимости от номера аргумента

switch (argoNumber) {

case 1:

{

alClock();

break;

}

case 2:

{

aNru();

break;

}

default:

break;

}

return 0;

}

1. **Выводы:**

Ознакомился с принципами организации виртуальной памяти.