**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 № 6

по дисциплине «Основы алгоритмизации и программирования»

Синхронизация процессов

Программирование планировщиков процессов

Выполнил студент

группы ИТИ-11

*Буякевич Д.О.*

Проверил преподаватель

*Карась О.В.*

Гомель 2022

**Цель работы**: создать функцию для выполнения дефрагментации файловой системы.

**Задание:** дополнить программу, разработанную при выполнении лабораторной работы №13 вспомогательной утилитой, согласно варианта (см. табл. 14).

|  |  |
| --- | --- |
| 3 | Дефрагментация |

**Ход работы:**

На рисунке 1 показано меню ЛБ.

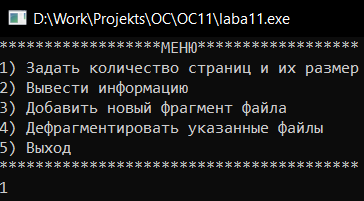
****

Рисунок 1 – Меню ЛБ

На рисунке 2 показаны ввод кол-ва страниц и размер

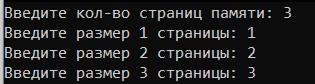


Рисунок 2 – Размер страниц

На рисунке 3 показан пункт 2 вывод информации о страницах

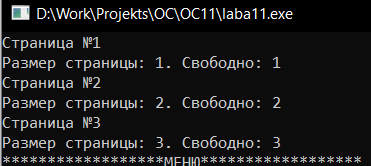


Рисунок 3 – Информация о страницах

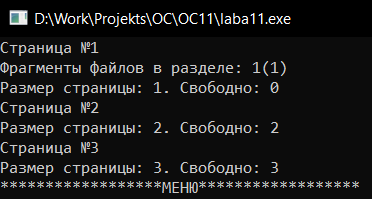
На рисунке 4 показан добавление фрагмента файла 

Рисунок 4 – Добавление фрагмента

На рисунке 5 показаны страницы до фрагментации

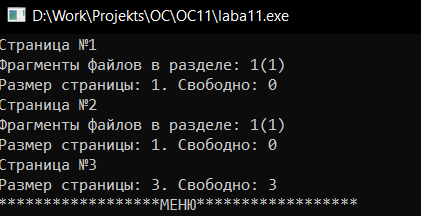


Рисунок 5 – До фрагментации

На рисунке 6 показана фрагментация

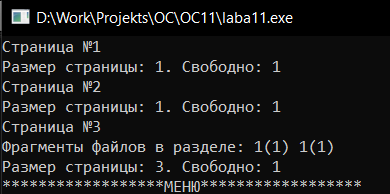


Рисунок 6 – Фрагментация

**Вывод:** в ходе данной лабораторной работы были изучены методы дефрагментации и реализован один из них.

**ПРИЛОЖЕНИЕ А**

**Листинг программы**

#include <locale.h>

#include <iostream>

#include <windows.h>

#include <vector>

#include <string>

#include <cstdlib>

#include <ctime>

using namespace std;

typedef struct {

int pageNum;

int isFreePage;

int pageMemory;

int pageFreeMemory;

} virtualPage;

typedef struct {

string fileName;

int fileMemory;

int filePage;

int isDef;

} virtualFile;

virtualPage \* page;

vector <virtualFile> files(1);

int fileNum = 0;

int pageNum = 0;

void inputData()

{

int N, i = 0;

cout << "Введите кол-во страниц памяти: ";

cin >> N;

page = new virtualPage[N];

while (i < N)

{

cout << "Введите размер " << i + 1 << " страницы: ";

cin >> page[i].pageMemory;

page[i].pageNum = i + 1;

page[i].isFreePage = 1;

page[i].pageFreeMemory = page[i].pageMemory;

i++;

}

pageNum = N;

}

void outputData()

{

int j, i = 0;

bool isFirst;

while (i < pageNum)

{

j = 0;

isFirst = true;

cout << "Страница №" << i + 1;

while (j < fileNum)

{

if (files[j].filePage == page[i].pageNum)

{

if (isFirst)

{

cout << "\nФрагменты файлов в разделе: ";

isFirst = false;

}

cout << files[j].fileName << "(" << files[j].fileMemory << ") ";

}

j++;

}

cout << "\nРазмер страницы: " << page[i].pageMemory << ". Свободно: " << page[i].pageFreeMemory << endl;

i++;

}

}

int menu()

{

int n;

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

printf("1) Задать количество страниц и их размер\n");

printf("2) Вывести информацию\n");

printf("3) Добавить новый фрагмент файла\n");

printf("4) Дефрагментировать файлы\n");

printf("5) Выход\n");

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

cin >> n;

return n;

};

void inputFile()

{

int i = 0, isPage, memory;

string name;

if (fileNum == 0)

{

cout << "Введите НАЗВАНИЕ файла: ";

cin >> name;

cout << "Введите НОМЕР нужной страницы: ";

cin >> isPage;

cout << "Введите РАЗМЕР фрагмента файла: ";

cin >> memory;

}

else

{

files.insert(files.end(), files[0]);

cout << "Введите НАЗВАНИЕ файла: ";

cin >> name;

cout << "Введите НОМЕР страницы: ";

cin >> isPage;

cout << "Введите РАЗМЕР фрагмента файла: ";

cin >> memory;

}

bool isFind = false;

while (memory > page[isPage - 1].pageFreeMemory && !isFind)

{

if (memory <= page[i].pageFreeMemory)

{

cout << "В выбранный раздел не вмещается файл такого размера, файл помещён в раздел №\n" << i + 1;

isPage = i + 1;

isFind = true;

}

i++;

if (i > pageNum)

{

cout << "Нет возможности разместить фрагмент в памяти";

return;

}

}

files[fileNum].fileName = name;

files[fileNum].filePage = isPage;

files[fileNum].fileMemory = memory;

files[fileNum].isDef = 0;

page[isPage - 1].pageFreeMemory -= memory;

page[isPage - 1].isFreePage = 0;

fileNum++;

}

void oneDefrag()

{

int i = 0, j = 0, k = 0, size, num, pg;

string name;

bool isFind;

while (i < fileNum)

{

j = 0;

k = 0;

name = files[i].fileName;

pg = files[i].filePage;

size = 0;

isFind = false;

while (j < fileNum)

{

if (name == files[j].fileName && !files[j].isDef)

{

size += files[j].fileMemory;

if (pg != files[j].filePage)

isFind = true;

}

j++;

}

j = 0;

if (isFind)

{

while (k < pageNum)

{

if (page[k].pageFreeMemory >= size)

{

while (j < fileNum)

{

if ( name == files[j].fileName && files[j].filePage != (k + 1) )

{

num = files[j].filePage - 1;

files[j].filePage = k + 1;

page[k].pageFreeMemory -= files[j].fileMemory;

files[j].isDef = 1;

page[num].pageFreeMemory += files[j].fileMemory;

}

j++;

}

}

k++;

}

}

i++;

}

}

int main()

{

setlocale(LC\_ALL, "RUS");

int n;

while ((n = menu()) != 5)

{

system("cls");

switch (n)

{

case 1: inputData();

break;

case 2: outputData();

break;

case 3: inputFile();

break;

case 4: oneDefrag();

break;

default:

break;

}

}

system("pause");

delete [] page;

return 0;

}

// 3 стр 1 1 3 добавите фрагмент 1 1 1 , 1 1 1