**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 № 11

по дисциплине «Операционные системы»

на тему: «Управление файловой системой»

Выполнил: студент гр. ИТП-11

Ермолин Д.К.

Принял: преподаватель-стажёр

Карась О.В.

Гомель 2022

**Цель:** изучить принципы работы файлового менеджера и утилиты для работы с ним.

**Задача:**

Разработать утилиту в соответствии со своим вариантом.

**Вариант 10:**

Дефрагментация.

**Задание**

Дефрагментация - процесс перераспределения фрагментов файлов и логических структур файловых систем на дисках для обеспечения непрерывной последовательности [кластеров](https://ru.wikipedia.org/wiki/%D0%9A%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80_(%D0%B5%D0%B4%D0%B8%D0%BD%D0%B8%D1%86%D0%B0_%D1%85%D1%80%D0%B0%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)).

При первом запуске программы пользователя просят ввести объем памяти. Далее открывается меню с основными функциями программы Пример главного меню программы указан на рисунке 1.

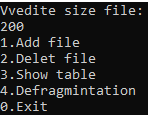


Рисунок 1 – Главное меню программы

Потом пользователь может создать несколько файлов. Там он указывает название и размер файла, который этот файл будет занимать на диске. Пример добавления нового файла указан на рисунке 2.

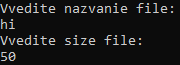


Рисунок 2 – Добавление файла

После пользователь может открыть таблицу памяти, где указаны названия файлов и их объем. Пример таблицы памяти указан на рисунке 3.

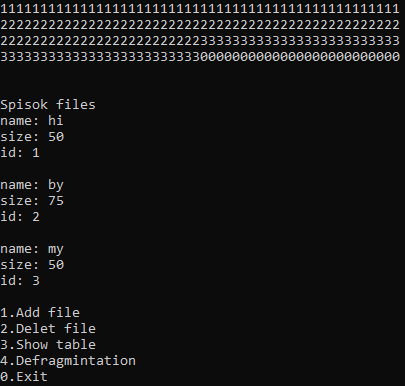


Рисунок 3 – Таблица памяти

Пользователь также может удалить файл. При выборе этого пункта будет открываться новое окно куда пользователь указывает название файла, который нужно удалить. Пример удаления файла указан на рисунке 4.



Рисунок 4 – Удаление файла

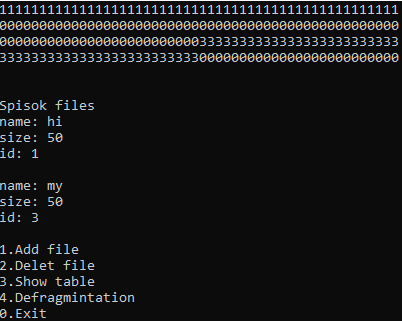


Рисунок 5 – Таблица памяти после удаления файла

Пример таблицы памяти после дефрагментации указан на рисунке 6.

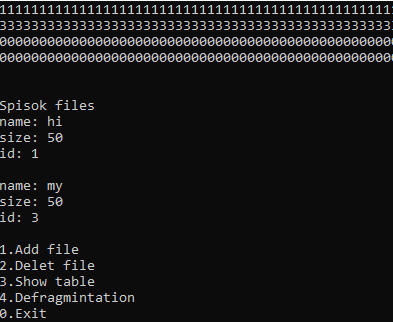


Рисунок 6 – Таблица памяти после дефрагментации

На механическом HDD этот процесс полезен еще и тем, что после него уменьшается количество необходимых для считывания информации передвижений головки по диску.

**Вывод**: в ходе лабораторной работы были изучены основные утилиты для работы с файловым менеджером и разработана программа дефрагментации на языке программирования *C*.

**Листинг программы**

#include <stdio.h>

#include <conio.h>

#include <windows.h>

#include <stdlib.h>

#include <string.h>

struct infoAboutFile

{

int size;

char name[100];

int id;

};

void addFile(struct infoAboutFile \*file, int \*memoryLeft, int \*numberOfFile);

void deletFile(struct infoAboutFile \*file, int \*memoryLeft, int \*numberOfFile);

void showTable(int size, struct infoAboutFile \*file, int numberOfFile, int memoryLeft);

void defragment(int size, struct infoAboutFile \*file, int \*numberOfFile, int \*memoryLeft);

void main()

{

SetConsoleCP (1251);

SetConsoleOutputCP (1251);

struct infoAboutFile files[1000];

int sizeOfMemory, k = 1, memoryLeft, numberOfFile = 0;

puts("Vvedite size file: ");

scanf("%d", &sizeOfMemory);

memoryLeft = sizeOfMemory;

while(k != 0)

{

puts("1.Add file");

puts("2.Delet file");

puts("3.Show table");

puts("4.Defragmintation");

puts("0.Exit");

printf("Your chose: ");

scanf("%d", &k);

system("cls");

switch(k)

{

case 1:

addFile(&files, &memoryLeft, &numberOfFile);

break;

case 2:

deletFile(&files, &memoryLeft, &numberOfFile);

break;

case 3:

showTable(sizeOfMemory, &files, numberOfFile, memoryLeft);

break;

case 4:

defragment(sizeOfMemory, &files, &numberOfFile, &memoryLeft);

break;

case 0:

break;

default:

break;

}

}

fflush(stdin);

getchar();

}

void addFile(struct infoAboutFile \*file, int \*memoryLeft, int \*numberOfFile)

{

puts("Vvedite nazvanie file: ");

scanf("%s", &(file+(\*numberOfFile))->name);

puts("Vvedite size file: ");

scanf("%d", &(file+(\*numberOfFile))->size);

if((file+(\*numberOfFile))->size > \*memoryLeft)

puts("Nedostatochno mesta");

else

{

(file+(\*numberOfFile))->id = \*numberOfFile + 1;

\*memoryLeft = \*memoryLeft - (file+(\*numberOfFile))->size;

\*numberOfFile += 1;

}

}

void deletFile(struct infoAboutFile \*file, int \*memoryLeft, int \*numberOfFile)

{

int i, k = 0;

char deletName[100];

puts("Vvedite nazvanie file: ");

scanf("%s", &deletName);

for(i = 0; i < \*numberOfFile; i++)

{

if(strcmp(deletName, (file+i)->name) == 0)

{

\*memoryLeft = \*memoryLeft + (file+i)->size;

(file+i)->id = 0;

k = 1;

}

}

if(k == 0)

puts("Takogo file net");

}

void showTable(int size, struct infoAboutFile \*file, int numberOfFile, int memoryLeft)

{

int i, j, k = 1, fullMemoryLeft;

fullMemoryLeft = memoryLeft;

for(j = 0; j < numberOfFile; j++)

{

if((file+j)->id == 0)

{

for(i = 0; i < (file+j)->size; i++)

{

printf("%d", (file+j)->id);

if(k % 50 == 0)

printf("\n");

k += 1;

fullMemoryLeft -= 1;

}

}

else

{

for(i = 0; i < (file+j)->size; i++)

{

printf("%d", (file+j)->id);

if(k % 50 == 0)

printf("\n");

k += 1;

}

}

}

for(i = 0; i < fullMemoryLeft; i++)

{

printf("%d", (file+j)->id);

if(k % 50 == 0)

printf("\n");

k += 1;

}

printf("\n\n");

printf("Spisok files\n");

for(i = 0; i < numberOfFile; i++)

{

if((file+i)->id != 0)

{

printf("name: %s\n", (file+i)->name);

printf("size: %d\n", (file+i)->size);

printf("id: %d\n\n", (file+i)->id);

}

}

}

void defragment(int size, struct infoAboutFile \*file, int \*numberOfFile, int \*memoryLeft)

{

int i, j;

i = 0;

while(i < \*numberOfFile)

{

if((file+i)->id == 0)

{

j = i;

while(j < \*numberOfFile)

{

\*(file+j) = \*(file+j+1);

j++;

}

\*numberOfFile -= 1;

i -= 1;

}

i +=1;

}

}