**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

Демиденко Д.В.

Принял: преподаватель

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 9:**

Дефрагментация.

**Задание**

Дефрагментация – процесс перераспределения фрагментов файлов на дисках, в результате которого данные перезаписывается в непрерывной области диска для более быстрого доступа к ним. Этот процесс приводит систему в порядок, находит и объединяет фрагментированные файлы на локальных томах, чтобы повысить производительность системы. Для реализации этой утилиты был написана программа на языке программирования *C*. При первом запуске программы пользователя просят ввести объем памяти. Далее открывается меню с основными функциями программы такие как: добавить файл, удалить файл, показать таблицу памяти и дефрагментация. Пример главного меню программы указан на рисунке 1.

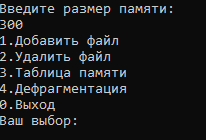


Рисунок 1 – Главное меню программы

В меню предлагается ввести название файла и объем памяти, который этот файл будет занимать на диске. Если объем памяти будет меньше, чем количество свободного места на диске то файл не будет добавлен. Пример добавления нового файла указан на рисунке 2.

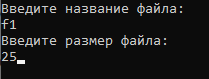


Рисунок 2 – Главное меню программы

После добавления нескольких файлов пользователь может открыть таблицу памяти, где указаны названия файлов и их объем. А сверху указана таблица памяти, где символы 0 это столько сколько осталось места на диске. Остальные цифры – это объем, который зан­­­­­­­­­имает каждый файл. Пример таблицы памяти указан на рисунке 3.

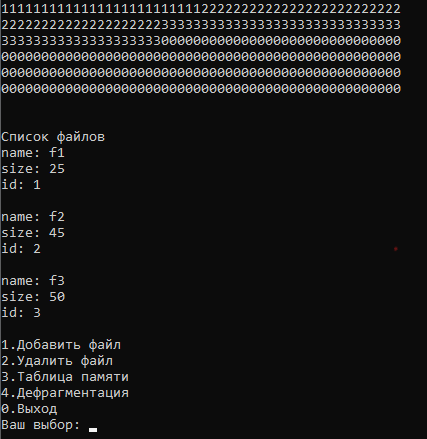


Рисунок 3 – Таблица памяти

Один из пунктов меню это удаление файла. При выборе этого пункта будет открываться новое окно куда пользователь указывает название файла, который нужно удалить. Если файла с таким названием нет, то выведет ошибку. Пример удаления файла указан на рисунке 4.



Рисунок 4 – ввод названия файла для удаления

После удаления файла таблица памяти будет изменена. Новая таблица указана на рисунке 5.

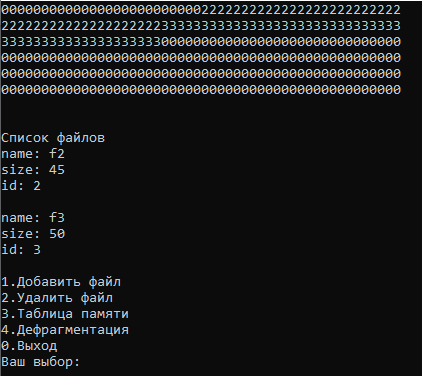


Рисунок 5 – Таблица памяти после удаления файла

Если компьютеру будет необходимо считать сначала первый, а потом и третий файл то магнитной головке придется делать лишние движения, что приведёт к её износу. Для решения этой проблемы была использована дефрагментация. Которая располагает все блоки памяти впритык друг к другу. Пример таблицы памяти после дефрагментации указан на рисунке 6.

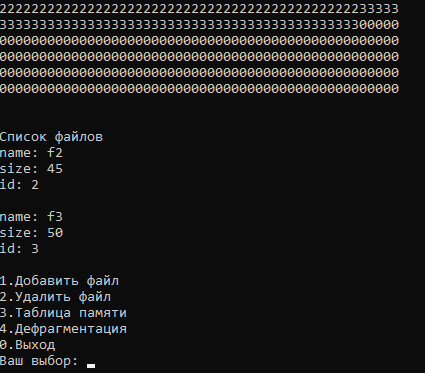


Рисунок 6 – Таблица памяти после дефрагментации

**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("Введите размер памяти: ");

scanf("%d", &sizeOfMemory);

memoryLeft = sizeOfMemory;

while(k != 0)

{

puts("1.Добавить файл");

puts("2.Удалить файл");

puts("3.Таблица памяти");

puts("4.Дефрагментация");

puts("0.Выход");

printf("Ваш выбор: ");

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("Введите название файла: ");

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

puts("Введите размер файла: ");

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

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

puts("Место недостаточно");

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("Введите название файла: ");

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("Такого файла нет");

}

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("Список файлов\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;

}

}