ПИК-2

Kursova rabota

Velko Milanov, 123220014, FKST, KSI, 79 grupa

Задание No 9 дисциплина: ПИК II – курсова задача

Да се разработи програма с меню тип файлов индексатор.

Индексатор на файлове пази списък с файловете на компютъра за по-бързото им търсене,

вместо да ги чете папка по папка от хард-диска. Той трябва да има:

1. Данните за индексираните файловете се съхраняват в двоични файл. Файла започва с броя файлове следван от

файловете. За всеки индексиран файл се пазят следните данни:

• Пълен път (без файла);

• Име на файл;

• Размер на файла;

• Дата на създаване на файла - записана във формата ДД.ММ.ГГГГ.;

• Дата на модифициране на файла - записана във формата ДД.ММ.ГГГГ.;

• Флагове: hidden, read-only, system file, executable

2. Да се извършват следните обработки:

а) Добавяне на нов файл;

б) Намиране на даден файл и извеждане на неговите данни;

в) По зададена от потребителя папка, показва колко файла има в нея и колко място заема тя на хардиска

(файловете и рекурсивно папките в нея).

3. Данните да се поддържат в динамична структура – динамично заделен масив или едносвързан списък в

оперативната памет на ПК.

/\* Velko Milanov, FKST, KSI, 1. kurs, 123220014, 79 grupa:

Да се разработи програма с меню тип файлов индексатор.

Индексатор на файлове пази списък с файловете на компютъра за по-бързото им търсене,

вместо да ги чете папка по папка от хард-диска. Той трябва да има:

1. Данните за индексираните файловете се съхраняват в двоични файл. Файла започва с броя файлове следван от

файловете. За всеки индексиран файл се пазят следните данни:

• Пълен път (без файла);

• Име на файл;

• Размер на файла;

• Дата на създаване на файла - записана във формата ДД.ММ.ГГГГ.;

• Дата на модифициране на файла - записана във формата ДД.ММ.ГГГГ.;

• Флагове: hidden, read-only, system file, executable

2. Да се извършват следните обработки:

а) Добавяне на нов файл;

б) Намиране на даден файл и извеждане на неговите данни;

в) По зададена от потребителя папка, показва колко файла има в нея и колко място заема тя на хардиска

(файловете и рекурсивно папките в нея).

3. Данните да се поддържат в динамична структура – динамично заделен масив или едносвързан списък в

оперативната памет на ПК.

\*/

**Functions:**

*File \*new\_item()* - Това е функция за добавяне на нов елемент към списъка, прости функции printf и scanf.

*void print(File\* root)* - Това е функция за показване на списъка на екрана, цикълът while се използва, докато текущият елемент е NULL или докато няма повече елементи в списъка и той се покаже с printf.

*File \*create\_queue\_list(int size)* - Това е функция, която създава списък с елементи, които потребителят добавя.

*void free\_list(File\* root)* - Това е функция за освобождаване на паметта, заета от списъка.

*void newFile(File \*item,File \*item2)* - Това е функция за добавяне на нов файл от потребителя.

*void findFile(File \*root,char search[50])* - Това е функция за намиране на елемент в списъка, т.е. намиране на файла и неговите характеристики с помощта на параметър, посочен от потребителя, в този случай името на файла. Параметърът се сравнява с всички имена на файлове и когато функцията съвпада, той изхвърля всички характеристики на този файл.

*void record(File \*root, FILE \*fp)* - Това е функция за записване на структура, т.е. списък с елементи, в двоичен файл.

*void byLocation(File \*root,char folder[50])* - Това е функция, която взема параметъра на файла за местоположение от потребителя и го сравнява с всички елементи и където се припокрива, изхвърля дали в тази папка има файлове, колко подпапки има и колко памет заема на харддиск.

*int main()* - Това е основната функция, в която използваме всички тези функции по-горе, за да ги съберем в едно цяло, програмата.

**Code:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct File

{

char location[100];

char name[50];

int size;

char created[50];

char modified[50];

char attributes[50];

struct File\* next;

};

typedef struct File File;

File \*new\_item()

{

File \*result=NULL;

result = (File\*)malloc(sizeof(File));

printf("New Item -> location = ");

scanf("%s",&result->location);

printf("New Item -> name = ");

scanf("%s",&result->name);

printf("New Item -> size = ");

scanf("%d",&result->size);

printf("New Item -> created = ");

scanf("%s",&result->created);

printf("New Item -> modified = ");

scanf("%s",&result->modified);

printf("New Item -> attributes = ");

scanf("%s",&result->attributes);

result->next = NULL;

return result;

}

void print(File\* root)

{

File\* curr\_item = root;

while(curr\_item != NULL)

{

printf("Item has location %s\n",curr\_item->location);

printf("Item has name %s\n",curr\_item->name);

printf("Item has size %d\n",curr\_item->size);

printf("Item has created %s\n",curr\_item->created);

printf("Item has modified %s\n",curr\_item->modified);

printf("Item has attributes %s\n",curr\_item->attributes);

curr\_item=curr\_item->next;

}

}

File \*create\_queue\_list(int size)

{

File \*res\_root = NULL;

File \*end\_item = NULL;

int i;

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

{

if(res\_root == NULL)

{

res\_root = new\_item();

end\_item = res\_root;

}

else

{

end\_item->next = new\_item();

end\_item = end\_item->next;

}

}

return res\_root;

}

void free\_list(File\* root)

{

File\* curr\_item = root;

while(root != NULL)

{

root = root->next;

free(curr\_item);

curr\_item= root;

}

}

void newFile(File \*item,File \*item2)

{

item2->next=item->next;

item->next = item2;

}

void findFile(File \*root,char search[50])

{

int check;

int i;

File\* curr\_item = root;

while(curr\_item != NULL)

{

check=strcmp(search, curr\_item->name);

if(check==0)

{

printf("Item has location %s\n",curr\_item->location);

printf("Item has name %s\n",curr\_item->name);

printf("Item has size %d\n",curr\_item->size);

printf("Item has created %s\n",curr\_item->created);

printf("Item has modified %s\n",curr\_item->modified);

printf("Item has attributes %s\n",curr\_item->attributes);

}

curr\_item=curr\_item->next;

}

}

void record(File \*root, FILE \*fp)

{

if((fp = fopen("index", "wb"))==NULL)

{

printf("Cannot open file. \n");

exit(1);

}

if(fwrite(&root, sizeof(File), 1, fp) != 1)

{

printf("Write error occured. \n");

exit(1);

}

fclose(fp);

}

void byLocation(File \*root,char folder[50])

{

int check,counter=0, sum=0, check2, sub=0;

int i;

File\* curr\_item = root;

while(curr\_item != NULL)

{

check=strcmp(folder, curr\_item->location);

if(check==0)

{

counter++;

sum=sum + curr\_item->size;

printf("Folder has file with name %s\n",curr\_item->name);

}

char \*token = strtok(curr\_item->location, "/");

check2=strcmp(folder, token);

if(check2==0 && check!=0)

{

sub++;

sum=sum + curr\_item->size;

}

curr\_item=curr\_item->next;

}

printf("Folder has %d subfolder\n", sub);

printf("Folder has %d files\n", counter);

printf("Folder has memory of %dMB\n", sum);

}

int main()

{

File \*root = NULL;

FILE \*fp;

FILE \*fp2;

char searchFile[50], searchFolder[50];

int find;

int n;

printf("How many files:\n");

scanf("%d", &n);

root=create\_queue\_list(n);

record(root, fp);

printf("Add new file:\n");

newFile(root, new\_item());

n++;

print(root);

printf("Search file by name:\n");

scanf("%s", &searchFile);

findFile(root, searchFile);

record(root, fp2);

printf("Search folder:\n");

scanf("%s", &searchFolder);

byLocation(root, searchFolder);

free\_list(root);

return 0;

}

**Control example:**

A picture containing text, screenshot, computer, indoor

Description automatically generated

A picture containing text, screenshot, computer, indoor

Description automatically generated