# ЦЕЛЬ РАБОТЫ

Изучение информационных структур ОС, получение навыков обработки массивов данных и отладки программ циклической структуры.

# ПОСТАНОВКА ЗАДАЧИ

1. Согласно варианту №7 разработать функцию сортировки включениями таблицы дескрипторов файлов. Ключом для сортировки является дата создания файла.
2. Таблицу дескрипторов представить массивом структур. Дескриптор должен содержать имя файла (не более 8 символов), тип файла (не более 3 символов), дату создания (в формате чч.мм.гг), количество обращений (целое число), размер файла (целое число), время последней модификации(в формате час.мин).
3. В главной программе предусмотреть ввод и вывод исходных данных и результатов, а так же обращение к функции сортировки. Глобальные переменные не использовать.

# ОПИСАНИЕ ИСПОЛЬЗУЕМЫХ СТРУКТУР ДАННЫХ

В программной реализации на языке C, каждый элемент массива имеет структуру вида:

struct fileDescription\_t{  
 char name[9];  
 char type[4];  
 struct tm creatDate;  
 size\_t useAmount;  
 size\_t fileSize;  
 struct tm lastModification;  
};

где поля createDate и lastModification представлены структурами стандартной библиотеки языка С - time.h.

# ОПИСАНИЕ АЛГОРИТМА РЕШЕНИЯ ЗАДАЧИ



Рисунок 1 – Структурная схема алгоритма сортировки массива методом прямых включений

# СПЕЦИФИКАЦИИ ПОДПРОГРАММ

При запуске программы вводится требуемое количество файлов, далее для заполнения данными массива структур используется функция filesDataRandomFill, которая в качестве параметров получает указатель на массив структур и размер данного массива. Функция заполняет массив дескрипторов файлов псевдослучайными значениями даты и времени, а так же заполняет остальные поля.

# ТЕКСТ ПРОГРАММЫ

#include <stdio.h>  
#include <stdlib.h>  
#include <time.h>  
#include <string.h>  
#include <ctype.h>  
  
struct fileDescription\_t{  
 char name[9];  
 char type[4];  
 struct tm creatDate;  
 size\_t useAmount;  
 size\_t fileSize;  
 struct tm lastModification;  
};  
  
void filesDataRandomFill(struct fileDescription\_t array[], size\_t arrSize){  
 for(size\_t i = 0; i < arrSize; ++i){  
  
 char temp[] = "File\_";  
 char buff[5];  
 itoa(i+1,buff,10);  
 strcat(temp,buff);  
 strcat(array[i].name, temp);  
  
 if (i%2 == 0){  
 strcpy(array[i].type, "txt");  
 } else {  
 strcpy(array[i].type, "bin");  
 }  
  
 time\_t tempTime = time(NULL);  
 struct tm \*tempTimeStruct = localtime(&tempTime);  
 tempTimeStruct->tm\_mday = rand() % 31 ;  
 tempTimeStruct->tm\_mon = rand() % 11;  
 tempTimeStruct->tm\_year = 110 + (rand() % (120-110));  
 array[i].creatDate = \*tempTimeStruct;  
 array[i].useAmount = (size\_t)rand() % 200;  
 array[i].fileSize = (size\_t)rand() % 11111;  
 array[i].lastModification.tm\_hour = tempTimeStruct->tm\_hour + i;  
 array[i].lastModification.tm\_min = tempTimeStruct->tm\_min + i;  
 }  
}

void fileDataPrint(struct fileDescription\_t temp){  
 printf("%9s | %7s | %2.2d.%2.2d.%4.4d | %8d | %4d | %2.2d:%2.2d \n",temp.name, temp.type,  
 temp.creatDate.tm\_mday,(temp.creatDate.tm\_mon + 1),(1900 + temp.creatDate.tm\_year), temp.useAmount, temp.fileSize,  
 temp.lastModification.tm\_hour, temp.lastModification.tm\_min);  
}  
  
void printArr(struct fileDescription\_t \*arr, size\_t arrSize){  
 printf("\nFILE NAME | EXTENSION | CREATE DATA | USES AMOUNT | SIZE | LAST MODF\n");  
 printf("-----------------------------------------------------------------------\n");  
 for (size\_t i = 0; i < arrSize; ++i){  
 fileDataPrint(arr[i]);  
 }  
}  
short int compareDate(struct tm date1, struct tm date2){  
 if ((date1.tm\_year == date2.tm\_year) && (date1.tm\_mon == date2.tm\_mon) && (date1.tm\_mday > date2.tm\_mday)){ return 1;  
 } else if((date1.tm\_year == date2.tm\_year) && (date1.tm\_mon > date2.tm\_mon)) { return 1;  
 } else if (date1.tm\_year > date2.tm\_year){ return 1;  
 } else{ return 0; }}  
  
struct fileDescription\_t \*sortTheArr(struct fileDescription\_t \*arr, size\_t arrSize){  
 struct fileDescription\_t temp;  
 for (size\_t i = 1; i < arrSize; ++i){  
 temp = arr[i];  
 size\_t j;  
 for (j = i; ((compareDate(arr[j-1].creatDate,temp.creatDate)) && (j>=1)); --j){ arr[j] = arr[j-1];  
 }  
 arr[j] = temp;  
 } return arr;}  
  
int main() {  
 printf("ENTER NUMBER OF FILES: ");  
 size\_t filesAmount = 0;  
 scanf("%d",&filesAmount);  
 struct fileDescription\_t \*mainArr = (struct fileDescription\_t\*) calloc(filesAmount,sizeof(struct fileDescription\_t));  
 filesDataRandomFill(mainArr, filesAmount);  
 printArr(mainArr,filesAmount);  
 sortTheArr(mainArr,filesAmount);  
 printArr(mainArr,filesAmount);  
 free(mainArr);   
 return 0;  
}

# ТЕСТИРОВАНИЕ ПРОГРАММЫ

Для полной проверки работоспособности программы было проведено 2 тестовых запуска, с разным количеством исходных данных:

На рисунке 2 изображены исходные данные и результат выполнения их сортировки.

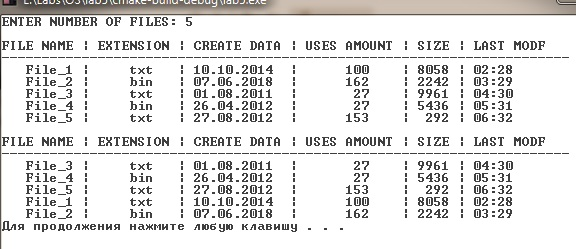


Рисунок 2 – Результат выполнения программы сортировки 5 элементов

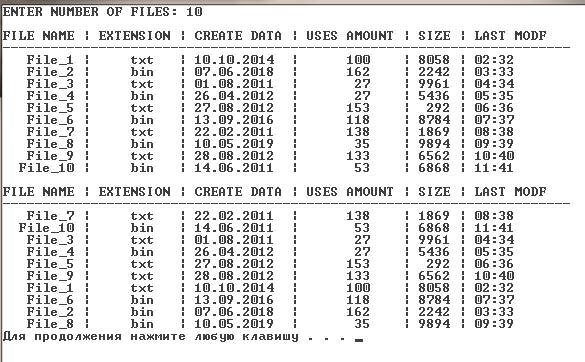


Рисунок 3 – Результат выполнения программы сортировки 10 элементов

ВЫВОД

При выполнении лабораторной работы была разработана, а так же протестирована программа, написанная на языке C, позволяющая сформировать и заполнить массив структур, содержащий дескрипторы файлов. Согласно варианту задания №7 была реализована функция сортировки массива структур методом прямых включений. Исходные данные и результат работы подпрограммы выводится в консоль.

По результатам тестирования, можно сделать вывод, что программа работоспособна и поставленная цель выполнена.