МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ
«Санкт-Петербургский национальный исследовательский
университет информационных технологий, механики и оптики»
Факультет информационных технологий и программирования
Кафедра информационных систем

Программирование
Лабораторная работа № 6

Выполнил студент:
Орлов Александр

Группа: М3107

САНКТ-ПЕТЕРБУРГ
2021

Целью лабораторной работы является разработка программы по архивированию
и распаковке нескольких файлов в один архив. Архиватор должен
1. Уметь архивировать несколько (один и более) указанных файлов в архив с
расширением *.arc
2. Уметь распаковывать файловых архив, извлекая изначально запаковонные
файлы
3. Предоставлять список файлов упакованных в архиве
Архиватор должен быть выполнен в виде консольного приложения, принимающего в качестве аргументов следующий параметры

- --file FILE

Имя файлового архива с которым будет работать архиватор

- --create

Команда для создания файлового архива

- --extract;

Команда для извлечения из файлового архива файлов

- --list

Команда для предоставления списка файлов, хранящихся в архиве

- FILE1 FILE2 .... FILEN

Свободные аргументы для передачи списка файлов для запаковки

`main.c:`

```c
/* Orlov Aleksandr, 12-5, M3107, 15.12.2021 */

#include <stdio.h>
#include <string.h>
#include "archive.h"

int main(int argc, char *argv[]) {
    FILE *f;
    unsigned char byte;
    char filename[MAX_STRING_SIZE];
    unsigned int sz;
    int files = 0, operation = 0;

    if (argc == 2 && !strcmp(argv[1], "--help")) {
        printf("This app is for creating and unpacking archives of files\n");
        printf("App supports 4 commands:\n--file - for the name of the
archive\n");
        printf("--create - for creating archive and for names of source
files\n");
        printf("--extract - for unpacking the archive\n");
        printf("--list - for displaying names of files in the archive\n");
        printf("Example of the command: Lab6.exe --file data.arc --create
D:\\a.txt C:/b.bin c.bmp\n");
        printf("PLEASE, BUY OUR LICENSE\n");
        return 0;
```

```
    }

    if (argc < 4) {
        printf("Error: incorrect number of arguments\n");
        return 1;
    }

    for (int i = 1; i < argc; i += 2) {
        if (!strcmp("--file", argv[i]))
            strcpy(filename, argv[i + 1]);
        else if (!strcmp("--create", argv[i])) {
            if (argc == 4) {
                printf("Error: incorrect flenames\n");
                return 1;
            }
            files = argc - 4, operation = 1;
            break;
        }
        else if (!strcmp("--extract", argv[i])) {
            operation = 2;
            break;
        }
        else if (!strcmp("--list", argv[i])) {
            operation = 3;
            break;
        }
    }

    if (operation == 0) {
        printf("Error: incorrect command\n");
        return 1;
    }

    switch (operation) {
        case 1:
            create_archive(argv, files, filename);
            break;
        case 2:
            unpack_archive(filename);
            break;
        case 3:
            file_list(filename);
            break;
    }

    return 0;
}
```

archive.c:

```
/* Orlov Aleksandr, 12-5, M3107, 15.12.2021 */

#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include "archive.h"

unsigned int fsize(FILE *fp){
    fseek(fp, 0L, SEEK_END);
    unsigned int sz=ftell(fp);
    fseek(fp,0L,SEEK_SET);
    return sz;
}
```

```
char *find_filename(char *filepath) {
    char *res = filepath;
    int filep_len = strlen(filepath);

    for (int i = filep_len - 1; i >= 0; i--)
        if (filepath[i] == '/' || filepath[i] == '\\') {
            res = (char *) malloc(filep_len - i);
            memcpy(res, filepath + i + 1, filep_len - 1 - i);
            res[filep_len - i - 1] = '\0';
            break;
        }
    return res;
}

int create_archive(char *argv[], int files, char *filename) {
    unsigned char files_number, byte;
    char *temp_filename, *text;
    FILE *output, *f;

    if (files > 255)
        return 0;

    output = fopen(filename, "wb");
    if (output == NULL)
        return 0;

    files_number = files;
    fwrite(&files_number, 1, 1, output);

    for (int i = 0; i < files; i++) {
        unsigned int file_size;

        f = fopen(argv[i + 4], "rb");
        temp_filename = find_filename(argv[i + 4]);
        if (f == NULL)
            continue;

        file_size = fsize(f);
        if (file_size > 4294967294) {
            fclose(f);
            continue;
        }
        if (strlen(temp_filename) > 255) {
            fclose(f);
            continue;
        }
        byte = strlen(temp_filename);

        /* Writing filename size byte */
        fwrite(&byte, 1, 1, output);

        /* Writing file size bytes */
        fwrite(&file_size, sizeof (unsigned int), 1, output);

        for (int j = 0; j < byte; j++)
            fwrite(&temp_filename[j], 1, 1, output);

        text = (char *) malloc(file_size);
        fread(text, 1, file_size, f);
        fwrite(text, 1, file_size, output);
        fclose(f);
        free(text);
    }
```

```
    fclose(output);
    return 1;
}

int unpack_archive(char *filename) {
    FILE *input;
    char files;

    input = fopen(filename, "rb");

    if (input == NULL)
        return 0;

    fread(&files, 1, 1, input);

    for (int i = 0; i < files; i++) {
        FILE *f;
        char filename_size;
        int file_size;
        char *temp_filename, *text;

        fread(&filename_size, 1, 1, input);
        fread(&file_size, 4, 1, input);

        temp_filename = (char *) malloc(filename_size + 1);
        fread(temp_filename, 1, filename_size, input);
        temp_filename[filename_size] = '\0';

        f = fopen(temp_filename, "wb");
        if (f == NULL)
            continue;

        text = (char *) malloc(file_size);
        fread(text, 1, file_size, input);

        fwrite(text, 1, file_size, f);
        fclose(f);
        free(temp_filename);
        free(text);
    }
    return 1;
}

void file_list(char *filename) {
    FILE *input;
    char files;

    input = fopen(filename, "rb");

    if (input == NULL)
        return;

    fread(&files, 1, 1, input);

    for (int i = 0; i < files; i++) {
        char filename_size;
        int file_size;
        char *temp_filename, *text;

        fread(&filename_size, 1, 1, input);
        fread(&file_size, 4, 1, input);

        temp_filename = (char *) malloc(filename_size + 1);
        fread(temp_filename, 1, filename_size, input);
```

```
        temp_filename[filename_size] = '\0';

        fseek(input, file_size, SEEK_CUR);

        printf("Filename %i: %s\n", i, temp_filename);
        free(temp_filename);
    }
}
```

archive.h:

```
/* Orlov Aleksandr, 12-5, M3107, 15.12.2021 */

#ifndef LAB6_ARCHIVE_H
#define LAB6_ARCHIVE_H

#define MAX_STRING_SIZE 512

int create_archive(char *argv[], int files, char *filename);
int unpack_archive(char *filename);
void file_list(char *filename);

#endif //LAB6_ARCHIVE_H
```

Вывод:

В результате выполнения лабораторной работы удалось реализовать программу для архивации произвольного количества (не больше 256) файлов. Кроме того, программа может выводить названия архивированных файлов и разархивировать файлы формата .arc.