**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования**

**«МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И ИНФОРМАТИКИ»**

**(МТУСИ)**

Кафедра «Информационная безопасность»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №0**

по дисциплине

**«Разработка безопасного ПО»**

на тему

«Разработка консольных приложений на языке программирования C»

**Вариант №10**

Выполнила:

студентка группы БПЗ1902

Клевцова В.А.

Проверил:

старший преподаватель кафедры ИБ

Барков В.В.

Москва 2022

**Цель работы**

* Овладеть навыками создания однофайловых и многофайловых проектов и научится создавать программы на языке C с применением арифметических операций, инструкций выбора, итерации, массивов, строк и структур.
* Овладеть навыками создания разветвляющихся алгоритмов на языке C с применением инструкций выбора, условных выражений, логических операций и операций отношения.
* Овладеть навыками создания циклических алгоритмов на языке C с применением инструкций цикла на примере алгоритмов работы с последовательностями чисел.
* Овладеть навыками работы с указателями и массивами в языке программирования C.
* Овладеть навыками работы со строками в языке программирования C.
* Овладеть навыками работы со структурами, объединениями, перечислениями и битовыми полями в языке C.
* Овладеть навыками выделения динамической памяти в языке программирования C.

**Задание к работе**

* *Задание 1*

Разработать функцию лексикографического сравнения двух строк. Функция должна иметь следующий прототип: int CompareStrings(const char \*lhs, const char \*rhs); Функция возвращает: 0, если строки равны; -1, если строка lhs предшествует строке rhs; 1, если строка rhs предшествует строке lhs.

* *Задание 2*

Разработать структуру Person, которая включает в себя следующие поля: Фамилия (строка), Имя (строка), Отчество (строка), Год рождения (беззнаковое число)

* *Задание 3*

Разработать перечисление SortType, включающее два значения: Asc – сортировка по возрастанию; Desc – сортировка по убыванию

Разработать функцию сортировки массива структур Person. Функция должна иметь следующий прототип: void Sort(struct Person \*persons, int count, enum SortType sortType); Функция сортирует массив persons по возрастанию или убыванию в зависимости от значения параметра sortType. Сортировка производится по следующим полям: Фамилия, Имя, Отчество, Возраст. Сравнение строк организовать с помощью функции, разработанной в задании 1.

* *Задание 4*

Разработать функцию, которая получает аргументы командной строки и создаёт на его основе динамический массив структур. Размер массива равен количеству аргументов -person в аргументах командной строки.

Интерпретация аргументов командной строки указана в таблице ниже.

|  |  |
| --- | --- |
| Название аргумента | Описание |
| -person | Создать новый элемент Person |
| --fn | Установить для текущего элемента Person поле Имя. Значение находится в следующем аргументе. По умолчанию «Не задано» |
| --ln | Установить для текущего элемента Person поле Фамилия |
| --mn | Установить для текущего элемента Person поле Отчество. По умолчанию «Не задано» |
| --age | Установить для текущего элемента Person поле Возраст. По умолчанию 18 |

* *Задание 5*

Разработать функцию, которая получает на вход массив структур Person и выводит его на экран в виде таблицы

* *Задание 6*

Разработать функцию main, которая вызывает функцию из задания 5 для создания массива стуктур Person из аргументов командной строки, запрашивает тип сортировки, сортирует массив с помощью функции из задания 3 и выводит на экран с помощью функции из задания 5.

* *Задание 7*

Скомпилировать разработанные в заданиях 1-6 исходные файлы. Заголовочные файлы не передаются на вход компилятору, а включаются препроцессором в ходе компиляции исходных файлов. Создать на основе откомпилированных файлов (объектных файлов) из заданий 1- 5 статическую библиотеку. Скомпоновать откомпилированный файл из задания 6 и статическую библиотеку в исполняемый файл.

**Выполнение работы**

*Ссылка на гитхаб:* *https://github.com/vikuliko/RBPOlab0*

**Задание 1.**

Файл CompareStrings.c

#include "Header.h"

int CompareStrings(const char \*lhs, const char \*rhs)

{

    int leftsize = strlen(lhs);

    int rightsize = strlen(rhs);

    for (int i = 0; i < leftsize && i < rightsize; i++)

        if (lhs[i] < rhs[i])

            return -1;

        else if (lhs[i] > rhs[i])

            return 1;

    if (leftsize < rightsize)

        return -1;

    else if (leftsize > rightsize)

        return 1;

    return 0;

}

**Задание 2**

Файл Header.h

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

int CompareStrings(const char \*lhs, const char \*rhs);

struct Person

{

  char \*fn,

      \*mn,

      \*ln;

  unsigned short age;

};

enum SortType

{

  Asc,

  Desc

};

void Sort(struct Person \*persons, int count, enum SortType sortType);

void CreatePersons(int argc, char \*argv[], struct Person persons[], int Counter);

void PrintPersons(struct Person persons[], int Counter);

**Задание 3**

#include "Header.h"

void Sort(struct Person \*persons, int count, enum SortType sortType)

{

    for (int min = 0; min < count - 1; min++)

    {

        int least = min;

        for (int j = min + 1; j < count; j++)

            if (CompareStrings(persons[j].fn, persons[least].fn) == -1)

                least = j;

            else if (CompareStrings(persons[j].fn, persons[least].fn) == 0)

                if (CompareStrings(persons[j].ln, persons[least].ln) == -1)

                    least = j;

                else if (CompareStrings(persons[j].ln, persons[least].ln) == 0)

                    if (CompareStrings(persons[j].mn, persons[least].mn) == -1)

                        least = j;

                    else if (CompareStrings(persons[j].mn, persons[least].mn) == 0)

                        if (persons[j].age <= persons[least].age)

                            least = j;

        struct Person tmp = persons[min];

        persons[min] = persons[least];

        persons[least] = tmp;

    }

    struct Person t;

    if (sortType == Desc)

        for (int i = 0; i < count / 2; i++)

        {

            t = persons[i];

            persons[i] = persons[count - i - 1];

            persons[count - i - 1] = t;

        }

}

**Задание 4**

#include "Header.h"

void CreatePersons(int argc, char \*argv[], struct Person persons[],

                   int Counter)

{

    for (int i = 0; i < Counter; i++)

    {

        persons[i].fn = malloc(sizeof(char) \* 100);

        strcpy(persons[i].fn, "");

        persons[i].mn = malloc(sizeof(char) \* 100);

        strcpy(persons[i].mn, "");

        persons[i].ln = malloc(sizeof(char) \* 100);

        strcpy(persons[i].ln, "");

        persons[i].age = 0;

    }

    for (int i = 1, count = 0; i < argc; i++)

    {

        if (CompareStrings(argv[i], "-person") == 0)

            count++;

        else if (CompareStrings(argv[i], "--fn") == 0 && argv[i + 1][0] != '-')

            strcpy(persons[count - 1].fn, argv[i + 1]);

        else if (CompareStrings(argv[i], "--mn") == 0 && argv[i + 1][0] != '-')

            strcpy(persons[count - 1].mn, argv[i + 1]);

        else if (CompareStrings(argv[i], "--ln") == 0 && argv[i + 1][0] != '-')

            strcpy(persons[count - 1].ln, argv[i + 1]);

        else if (CompareStrings(argv[i], "--age") == 0 && argv[i + 1][0] != '-')

            persons[count - 1].age = atoi(argv[i + 1]);

    }

}

**Задание 5**

#include "Header.h"

void PrintPersons(struct Person persons[], int Counter)

{

    printf("|--------------------|--------------------|--------------------|--------------------|\n");

    printf("|%20s|%20s|%20s|%20s|\n", "Last name", "First name", "Middle name", "Age");

    for (int i = 0; i < Counter; i++)

    {

        printf("|--------------------|--------------------|--------------------|--------------------|\n");

        printf("|%20s|%20s|%20s|%20d|\n", persons[i].fn, persons[i].ln,

               persons[i].mn, persons[i].age);

    }

    printf("|--------------------|--------------------|--------------------|--------------------|\n");

}

**Задание 6**

#include "Header.h"

int main(int argc, char \*argv[])

{

    int Counter = 0;

    for (int i = 1; i < argc; i++)

        if (CompareStrings(argv[i], "-person") == 0)

            Counter++;

    struct Person persons[Counter];

    CreatePersons(argc, argv, persons, Counter);

    printf("%s\n",

           "Choose the way to sort: ascending (1) or descending (0)");

    fflush(NULL);

    char ans[100];

    scanf\_s("%s", ans);

    if (CompareStrings(ans, "1") == 0)

        Sort(persons, Counter, Asc);

    else if (CompareStrings(ans, "0") == 0)

        Sort(persons, Counter, Desc);

    else

    {

        printf("%s\n%s\n", "Wrong input", "Stopping the program...");

        return 0;

    }

    PrintPersons(persons, Counter);

    system("pause");

    return 0;

}

**Файл Makefile:**

CC=gcc

MAIN\_SOURCES=Main.c

MAIN\_OBJECTS=$(MAIN\_SOURCES:.c=.o)

LIB=funcs.a

LIB\_SOURCES=CompareStrings.c CreatePersons.c PrintPersons.c Sort.c

LIB\_OBJECTS=$(LIB\_SOURCES:.c=.o)

DEPS=Header.h

fill:

    ./app -person --fn A --ln A --mn A --age 10 -person --fn A --ln A --mn A --age 20 -person --fn A --ln A --mn C --age 10 -person --fn A --ln B --mn A --age 15 -person --fn A --ln B --mn C --age 25

all: app

app: $(LIB) $(MAIN\_OBJECTS)

    $(CC) -lm -o $@ $(MAIN\_OBJECTS) $<

funcs.a: $(LIB\_OBJECTS)

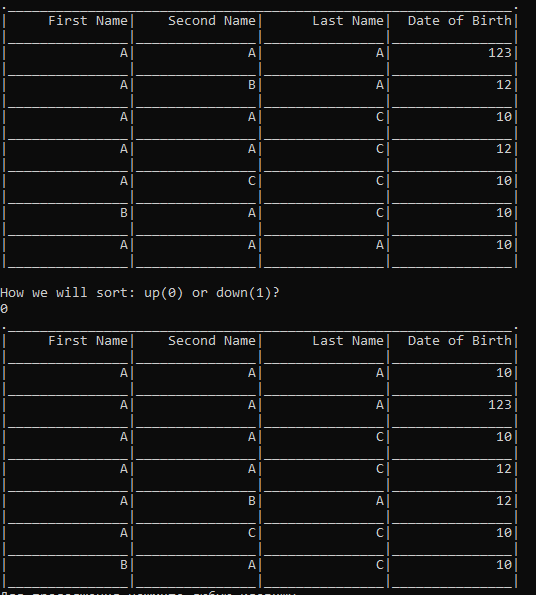
    ar rcs $@ $^

libs: $(LIB)

clean:

    del \*.o, \*.a, \*.exe

**Результат выполнения программы:**

****

