Лабораторная 5

Задание:разделите программу лабораторной 1 на модули. Реализуйте функции работы со списками в библиотеке совместного доступа .so

Цель:получение навыков разработки библиотек динамической компоновки на платформе Linux.

student_list.h, содержит определения структуры Student и прототипы функций

```
#ifndef STUDENT_LIST_H
#define STUDENT_LIST_H

#include <stdio.h>

#define MAX_NAME_LENGTH 30

struct Student {
  struct Student *next;
  char surname[MAX_NAME_LENGTH];
  double mark;
};

void addStudent(const char surname[], double mark);
void printList();
void serialize();
void deserialize();
void searchInList(const char surname[]);
void deleteFromList(const char surname[]);
void sortByMark();

#endif  // STUDENT_LIST_H
```

student_list.c, в котором реализуются функции, определенные в заголовочном файле

```
#include "student_list.h"

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct Student *head = NULL;
struct Student *tail = NULL;

void addStudent(const char surname[], double mark) {
  struct Student *newNode = (struct Student *)malloc(sizeof(struct
Student));
  if (newNode == NULL) {
    fprintf(stderr, "Memory allocation failed\n");
    return;
```

```
  }
  strncpy(newNode->surname, surname, MAX_NAME_LENGTH);
  newNode->mark = mark;
  newNode->next = NULL;

  if (head == NULL) {
    head = newNode;
    tail = newNode;
  } else {
    tail->next = newNode;
    tail = newNode;
  }
}

void printList() {
  struct Student *current = head;

  if (current == NULL) {
    printf("The list is empty.\n");
    return;
  }
  while (current != NULL) {
    printf("Surname: %s, Mark: %.2lf\n", current->surname,
current->mark);
    current = current->next;
  }
}
void serialize() {
  FILE *file = fopen("students.dat", "wb");
  if (file == NULL) {
    fprintf(stderr, "Error opening file for writing\n");
    return;
  }

  struct Student *current = head;
  while (current != NULL) {
    fwrite(current->surname, sizeof(current->surname), 1, file);
    fwrite(&(current->mark), sizeof(current->mark), 1, file);
    current = current->next;
  }
  fclose(file);
}
```

```
void deserialize() {
  FILE *file = fopen("students.dat", "rb");
  if (file == NULL) {
    fprintf(stderr, "Error opening file for reading\n");
    return;
  }

  char surname[MAX_NAME_LENGTH];
  double mark;

  while (fread(surname, sizeof(surname), 1, file) &&
         fread(&mark, sizeof(mark), 1, file)) {
    addStudent(surname, mark);
  }

  fclose(file);
}

void searchInList(const char surname[]) {
  struct Student *current = head;

  while (current != NULL) {
    if (strcmp(current->surname, surname) == 0) {
      printf("Found: Surname: %s, Mark: %.2lf\n", current->surname,
             current->mark);
      return;
    }
    current = current->next;
  }

  printf("The surname %s was not found.\n", surname);
}

void deleteFromList(const char surname[]) {
  struct Student *current = head;
  struct Student *prev = NULL;

  while (current != NULL) {
    if (strcmp(current->surname, surname) == 0) {
      if (prev == NULL) {
        head = current->next;
        if (head == NULL) {
          tail = NULL;
```

```
        }
      } else {
        prev->next = current->next;
        if (current == tail) {
          tail = prev;
        }
      }
      free(current);
      printf("Deleted: %s\n", surname);
      return;
    }
    prev = current;
    current = current->next;
  }

  printf("The surname %s was not found for deletion.\n", surname);
}

void sortByMark() {
  int swapped;

  do {
    swapped = 0;
    struct Student *ptr1 = head;

    while (ptr1 != NULL && ptr1->next != NULL) {
      if (ptr1->mark > ptr1->next->mark) {
        double tempMark = ptr1->mark;
        ptr1->mark = ptr1->next->mark;
        ptr1->next->mark = tempMark;

        char tempSurname[MAX_NAME_LENGTH];
        strcpy(tempSurname, ptr1->surname);
        strcpy(ptr1->surname, ptr1->next->surname);
        strcpy(ptr1->next->surname, tempSurname);

        swapped = 1;
      }
      ptr1 = ptr1->next;
    }

  } while (swapped);
}
```

main.c функция входа в программу

```c
#include <stdio.h>
#include <stdlib.h>

#include "student_list.h"

int main() {
  addStudent("Ivanov", 4.5);
  addStudent("Petrov", 3.0);
  addStudent("Sidorov", 5.0);

  printf("Before sorting:\n");
  printList();

  serialize();

  sortByMark();

  printf("\nAfter sorting:\n");
  printList();

  char searchSurname[MAX_NAME_LENGTH];
  printf("\nEnter the surname to search: ");
  scanf("%s", searchSurname);

  searchInList(searchSurname);

  char deleteSurname[MAX_NAME_LENGTH];
  printf("\nEnter the surname to delete: ");
  scanf("%s", deleteSurname);

  deleteFromList(deleteSurname);

  printf("\nAfter deletion:\n");
  printList();

  // deserialize();

  return 0;
}
```

Результат работы программы:

```
tania@TaniaLaptop:~/5sem/os/os_labs/lab5$ gcc -shared -o liblist.so student_list.c -fPIC
tania@TaniaLaptop:~/5sem/os/os_labs/lab5$ ls
 liblist.so   main.c   student_list.c   student_list.h  'Ла65-2023 .pdf'
tania@TaniaLaptop:~/5sem/os/os_labs/lab5$ gcc -o main main.c -L. -llist
tania@TaniaLaptop:~/5sem/os/os_labs/lab5$ ./main
Before sorting:
Surname: Ivanov, Mark: 4.50
Surname: Petrov, Mark: 3.00
Surname: Sidorov, Mark: 5.00

After sorting:
Surname: Petrov, Mark: 3.00
Surname: Ivanov, Mark: 4.50
Surname: Sidorov, Mark: 5.00

Enter the surname to search: Ivanov
Found: Surname: Ivanov, Mark: 4.50

Enter the surname to delete: Sidorov
Deleted: Sidorov

After deletion:
Surname: Petrov, Mark: 3.00
Surname: Ivanov, Mark: 4.50
tania@TaniaLaptop:~/5sem/os/os_labs/lab5$
```

-shared: Этот флаг указывает компилятору создать динамическую библиотеку (shared object). Библиотека будет иметь расширение .so, что обозначает, что она может быть использована несколькими программами одновременно.

-fPIC: Этот флаг означает "Position Independent Code" (код независимый от адреса). Он необходим для создания динамических библиотек, так как позволяет коду быть загруженным в любое место в памяти без необходимости изменения адресов. Это важно для совместного использования библиотек между разными процессами.

-L.: Этот флаг сообщает компилятору искать библиотеки в текущем каталоге

-lstudentlist: Этот флаг указывает компилятору подключить библиотеку с именем libstudentlist