Московский Авиационный Институт
(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

**Лабораторная работа №5 по курсу**
**«Операционные системы»**

# ДИНАМИЧЕСКИЕ БИБЛИОТЕКИ

Студент: Злобина Валерия Вадимовна
Группа: М8О–208Б–21
Вариант: 20
Преподаватель: Соколов Андрей Алексеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2022.

**Цель работы**
Целью является приобретение практических навыков в:

- Создание динамических библиотек
- Создание программ, которые используют функции динамических библиотек

**Задание**
Требуется создать динамические библиотеки, которые реализуют определенный функционал.
Далее использовать данные библиотеки 2-мя способами:
Во время компиляции (на этапе «линковки»/linking)
Во время исполнения программы. Библиотеки загружаются в память с помощью интерфейса ОС для работы с динамическими библиотеками
В конечном итоге, в лабораторной работе необходимо получить следующие части: Динамические библиотеки, реализующие контракты, которые заданы вариантом; Тестовая программа (программа №1), которая используют одну из библиотек, используя знания полученные на этапе компиляции;
Тестовая программа (программа №2), которая загружает библиотеки, используя только их местоположение и контракты.
Провести анализ двух типов использования библиотек.

**Вариант 36**
**Функция 1**
Отсортировать целочисленный массив

- Пузырьковая сортировка
- Сортировка Хоара

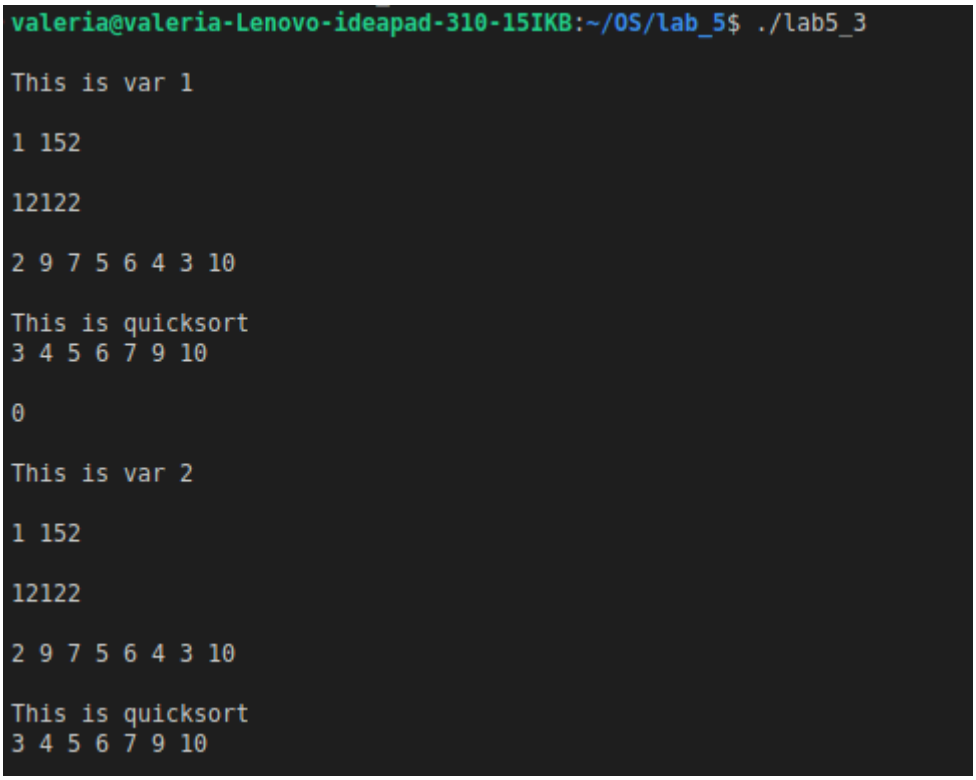
**Функция 2**
Перевод числа x из десятичной системы счисления в другую

- Другая система счисления двоичная
- Другая система счисления троичная

## Общие сведения о программе

Программа состоит из четырёх файлов: 1.c и 2.c — содержат в себе реализацию функций двумя различными способами, main.c — использует библиотеки на этапе компиляции, main2.c — загружает библиотеки, используя их местоположение и контракты.

Пример запуска программы:




## Исходный код:

## main.c

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

#define STR_MAX 32
#define MAX_SIZE 128

char* transfer(int);
void sort(int*, int);

int read_number(int *result) {
    char *buf = calloc(sizeof(char), STR_MAX);
    char c = 0;
    short i = 0;

    if (read(0, &c, 1) < 0) {
        exit(1);
    }
    if ((c == ' ') || (c == '\n') || (c == '\0'))
        return 2;
    while (c != ' ' && c != '\n') {
        buf[i++] = c;
        read(0, &c, 1);
    }
    *result = atoi(buf);
```

```
    free(buf);
    if (c == '\n') {
        return 1;
    }
    return 0;
}


int main() {
    printf("Choose command \n 1 - is for changing the excision system \n 2 - is for sorting\n");

    int command;
    read_number(&command);

    while (command != -1) {
        switch (command) {
            case 1:
                int t;
                scanf("%d", &t);
                char *result = transfer(t);
                printf("\n%s\n\n", result);
                break;
            case 2:
                int c, last;
                int i = 0;
                int *a = malloc(MAX_SIZE * sizeof(int));
                do {
                    last = read_number(&c);
                    if (last == 2) {
                        break;
                    }
                    a[i++] = c;
                } while (last == 0);
                a = realloc(a, i * sizeof(int));
                sort(a, i);
                for (int j = 0; j < i; ++j) {
                    printf("%d ", a[j]);
                }
                printf("\n\n");
                break;
            default:
                printf("unknown command: %d\n", command);
                break;
        }
        read_number(&command);
    }
    return 0;
}
```

## main2.c

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <unistd.h>
#include <dlfcn.h>


#define STR_MAX 32
```

```
#define MAX_SIZE 128

int read_number(int *result) {
    char *buf = calloc(sizeof(char), STR_MAX);
    char c = 0;
    short i = 0;

    if (read(0, &c, 1) < 0) {
        exit(1);
    }
    if ((c == ' ') || (c == '\n') || (c == '\0'))
        return 2;
    while (c != ' ' && c != '\n') {
        buf[i++] = c;
        read(0, &c, 1);
    }
    *result = atoi(buf);
    free(buf);
    if (c == '\n') {
        return 1;
    }
    return 0;
}

int main() {
    int var = 1;
    void *handle = dlopen("./liblib1.so", RTLD_LAZY);
    char*(*transfer)(int) = dlsym(handle, "transfer");
    void(*sort)(int*, int) = dlsym(handle, "sort");

    printf("\nThis is var %d\n\n", var);

    int command;
    read_number(&command);

    while (command != -1) {
        switch (command) {
            case 0:
                if (var == 1) {
                    dlclose(handle);
                    var = 2;
                    handle = dlopen("./liblib2.so", RTLD_LAZY);
```

```
            transfer = dlsym(handle, "transfer");
            sort = dlsym(handle, "sort");
        }
        else {
            dlclose(handle);
            var = 1;
            handle = dlopen("./liblib1.so", RTLD_LAZY);
            transfer = dlsym(handle, "transfer");
            sort = dlsym(handle, "sort");
        }
        printf("\nThis is var %d\n\n", var);
        break;
    case 1:
        int t;
        scanf("%d", &t);
        char *result = (*transfer)(t);
        printf("\n%s\n\n", result);
        break;
    case 2:
        int c, last;
        int i = 0;
        int *a = malloc(MAX_SIZE * sizeof(int));
        do {
            last = read_number(&c);
            if (last == 2) {
                break;
            }
            a[i++] = c;
        } while (last == 0);
        a = realloc(a, i * sizeof(int));
        (*sort)(a, i);
        for (int j = 0; j < i; ++j) {
            printf("%d ", a[j]);
        }
        printf("\n\n");
        break;
    default:
        printf("unknown command: %d\n", command);
        break;
  }
  read_number(&command);
}
```

```
    dlclose(handle);
    return 0;
}
```

## 1.c

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

#define STR_MAX 32

char* transfer(int a) {   // to binary
    char *temp = calloc(sizeof(char), STR_MAX);
    int sign = (a >= 0) ? 0 : 1;
    int b = (a > 0) ? a : -a;
    int i = 0;
    while (b != 0) {
        temp[i++] = '0' + (b % 2);
        b /= 2;
    }
    char *result = calloc(sizeof(char), i + sign);
    for (int j = 0; j < i; ++j) {
        result[j + sign] = temp[i - j - 1];
    }
    if (sign == 1) {
        result[0] = '-';
    }
    free(temp);
    return(result);

}

void sort(int *a, int n) {
    printf("\nThis is bubble sort\n");
    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (a[j + 1] < a[j]) {
                int k = a[j];
                a[j] = a[j + 1];
                a[j + 1] = k;
            }
```

```
        }
    }
}
```

## 2.c

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

#define STR_MAX 32

char* transfer(int a) {    // to ternary
    char *temp = calloc(sizeof(char), STR_MAX);
    int sign = (a >= 0) ? 0 : 1;
    int b = (a > 0) ? a : -a;
    int i = 0;
    while (b != 0) {
        temp[i++] = '0' + (b % 3);
        b /= 3;
    }
    char *result = calloc(sizeof(char), i + sign);
    for (int j = 0; j < i; ++j) {
        result[j + sign] = temp[i - j - 1];
    }
    if (sign == 1) {
        result[0] = '-';
    }
    free(temp);
    return(result);

}

void quicksort(int* a, int f, int l) {
    int m = a[(f + l) / 2];
    int i = f, j = l;
    do {
        while(a[i] < m) {
            i++;
        }
        while(a[j] > m) {
            j--;
```

```
        }
        if (i <= j) {
            if (i < j) {
                int temp = a[i];
                a[i] = a[j];
                a[j] = temp;
            }
            i++;
            j--;
        }
    } while(i <= j);
    if (i < l) {
        quicksort(a, i, l);
    }
    if (j > f) {
        quicksort(a, f, j);
    }
}

void sort(int *a, int n) {
    printf("\nThis is quicksort\n");
    quicksort(a, 0, n - 1);
}
```

**Общий метод и алгоритм решения**

В библиотеке 1.c реализованы перевод из десятичной системы в двоичную и пузырьковая сортировка, в библиотеке 2.c реализованы перевод из десятичной системы в троичную и сортировка Хоара.

Последовательность команд для компиляции:

```
gcc -fPIC -c 1.c
gcc -fPIC -c 2.c
gcc -shared -o liblib1.so 1.o
gcc -shared -o liblib2.so 2.o
gcc main.c -c
gcc -o lab5_1 main.o -L. -llib1 -Wl,-rpath,.
gcc -o lab5_2 main.o -L. -llib2 -Wl,-rpath,.
```

**Вывод**

Проделав лабораторную работу, я приобрёл практические навыки, необходимые для работы с динамическими библиотеками.