**Московский авиационный институт (национальный исследовательский университет)**

Институт информационных технологий и прикладной математики

«Кафедра вычислительной математики и программирования»

**Лабораторная работа по предмету "Операционные системы" №4**

Студент: Федоров М. А.

Преподаватель: Миронов Е.С.

Группа: М8О-207Б-22

Дата: 15.09.2022

Оценка:

Подпись:

Оглавление

[Цель работы 3](#__RefHeading__738_2004454639)

[Постановка задачи 3](#__RefHeading__740_2004454639)

[Общие сведения о программе 4](#__RefHeading__742_2004454639)

[Общий алгоритм решения 5](#__RefHeading__744_2004454639)

[Реализация 5](#__RefHeading__746_2004454639)

[Пример работы 8](#__RefHeading__748_2004454639)

[Вывод 9](#__RefHeading__750_2004454639)

# **Цель работы**

Целью является приобретение практических навыков в:

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

# **Постановка задачи**

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 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.

Пользовательский ввод для обоих программ должен быть организован следующим образом:

* Если пользователь вводит команду «0», то программа переключает одну реализацию контрактов на другую (необходимо только для программы №2). Можно реализовать лабораторную работу без данной функции, но максимальная оценка в этом случае будет «хорошо»;
* «1 arg1 arg2 … argN», где после «1» идут аргументы для первой функции, предусмотренной контрактами. После ввода команды происходит вызов первой функции, и на экране появляется результат её выполнения;
* «2 arg1 arg2 … argM», где после «2» идут аргументы для второй функции, предусмотренной контрактами. После ввода команды происходит вызов второй функции, и на экране появляется результат её выполнения.

*Вариант 29:*

**Контракты и реализации функций**

|  |  |  |  |
| --- | --- | --- | --- |
| Рассчет значения числа Пи при заданной длине ряда (K) | float Pi(int K) | Ряд Лейбница | Формула Валлиса |
| Перевод числа x из десятичной системы счисления в другую | Char\* translation(long x) | Другая система счисления двоичная | Другая система счисления троичная |

# **Общие сведения о программе**

Программа представлена пятью файлами **–** lib.h, lib\_first.c, lib\_second.c, program\_first.c и program\_second.c.

В программе используются следующие системные вызовы:

**scanf() –** считывает строку из входного потока.

**dlopen() –** подключает динамическую библиотеку и возвращает ее дескриптор

**dlsym()** **–** поиск адреса начала указанной строки в заданной библиотеке и возвращает указатель на этот адрес.

# **Общий алгоритм решения**

# **Реализация**

**lib.h**

#ifndef LIB\_H

#define LIB\_H

float pi(int );

char\* translation(long );

#endif

**-----------------------------------------------------------**

**lib\_first.c**

#include "lib.h"

#include <stdlib.h>

float pi(int row\_order){    //Ряд Лейбница

    float PI=0.0;

    int sign=1;

    for (size\_t iteration\_row = 0; iteration\_row < row\_order; ++iteration\_row)

    {

        PI+=(4.0/(2\*iteration\_row+1)\*sign);

        sign\*=-1;

    }

    return PI;

}

char\* translation(long x) {     //перевод в двоичную систему счисления

    int max\_binary\_length = 8 \* sizeof(long) + 1;

    char\* binary\_number = (char\*)malloc(max\_binary\_length);

    int index = max\_binary\_length - 1;

    binary\_number[index--] = '\0';

    if (x == 0) {

        binary\_number[index] = '0';

        return binary\_number;

    }

    while (x != 0) {

        int remainder = x % 2;

        binary\_number[index--] = remainder + '0';

        x /= 2;

    }

    return &binary\_number[index + 1];

}

**-----------------------------------------------------------**

**Lib\_second.c**

#include "lib.h"

#include <stdlib.h>

float pi(int row\_order){    // Формула Валисса

    float PI = 2.0;

    for (size\_t iteration\_row = 1; iteration\_row <= row\_order; ++iteration\_row)

    {

        PI \*= (4.0 \* iteration\_row \* iteration\_row) / ((4.0 \* iteration\_row \* iteration\_row) - 1);

    }

    return PI;

}

char \*translation(long x)

{

    int max\_ternary\_length = 8 \* sizeof(long) + 1;

    char \*ternary\_number = (char \*)malloc(max\_ternary\_length);

    int index = max\_ternary\_length - 1;

    ternary\_number[index--] = '\0';

    if (x == 0)

    {

        ternary\_number[index] = '0';

        return ternary\_number;

    }

    while (x != 0)

    {

        int remainder = x % 3;

        ternary\_number[index--] = remainder + '0';

        x /= 3;

    }

    return &ternary\_number[index + 1];

}

**-----------------------------------------------------------**

**Program\_first.c**

    #include "lib.h"

    #include <stdio.h>

    int main()

    {

        int request = 0;

        printf("requests:\n");

        printf("\"1\" - calculate PI \n");

        printf("\"2\" - converting a number to binary \n");

        printf("\"3\" - complete program \n");

        printf("Enter request: ");

        while (scanf("%d", &request) != EOF)

        {

            if (request == 1)

            {

                int order\_row;

                printf("Enter order\_row: ");

                scanf("%d", &order\_row);

                printf("number PI: %f\n", pi(order\_row));

            }

            else if (request == 2)

            {

                int decimal\_number;

                printf("Enter decimal\_number: ");

                scanf("%d", &decimal\_number);

                char \*binary\_number = translation(decimal\_number);

                printf("binary representation: %s\n", binary\_number);

            }

            else if (request == 3)

            {

                printf("The program is completed.\n");

                break;

            }

            else

            {

                printf("entered the wrong request.\n");

            }

            printf("Enter request: ");

        }

        return 0;

    }

**-----------------------------------------------------------**

**Program\_second.c**

#include "lib.h"

#include <stdio.h>

#include <stdbool.h>

#include <stdlib.h>

#include <dlfcn.h>

int main()

{

    float (\*pi\_function)(int row\_order) = NULL;

    char \*(\*translation\_function)(long x) = NULL;

    void \*library\_descriptor\_first = NULL;

    void \*library\_descriptor\_second = NULL;

    printf("requests:\n");

    printf("\"0\" - swap library \n");

    printf("\"1\" - calculate pi \n");

    printf("\"2\" - converting a number to binary/ternary \n");

    printf("\"3\" - complete program \n");

    printf("Enter request: ");

    bool is\_first\_library = true;

    int request = 0;

    if ((library\_descriptor\_first = dlopen("libLib1.so", RTLD\_LAZY)) == 0)

    {

        printf("OPEN LIBRARY ERROR\n");

        exit(-1);

    }

    if ((library\_descriptor\_second = dlopen("libLib2.so", RTLD\_LAZY)) == 0)

    {

        printf("OPEN LIBRARY ERROR\n");

        exit(-1);

    }

    pi\_function = dlsym(library\_descriptor\_first, "pi");

    translation\_function = dlsym(library\_descriptor\_first, "translation");

    while (scanf("%d", &request) != EOF)

    {

        if (request == 0)

        {

            if (is\_first\_library)

            {

                pi\_function = dlsym(library\_descriptor\_second, "pi");

                translation\_function = dlsym(library\_descriptor\_second, "translation");

                is\_first\_library=false;

            }

            else

            {

                pi\_function = dlsym(library\_descriptor\_first, "pi");

                translation\_function = dlsym(library\_descriptor\_first, "translation");

                is\_first\_library=true;

            }

        }

        else if (request == 1)

        {

            int order\_row;

            printf("Enter order\_row: ");

            scanf("%d", &order\_row);

            printf("number pi\_function: %f\n", (\*pi\_function)(order\_row));

        }

        else if (request == 2)

        {

            int decimal\_number;

            printf("Enter decimal\_number: ");

            scanf("%d", &decimal\_number);

            char \*number = (\*translation\_function)(decimal\_number);

            printf("number representation: %s\n", number);

        }

        else if (request == 3)

        {

            printf("The program is completed.\n");

            break;

        }

        else

        {

            printf("entered the wrong request.\n");

        }

        printf("Enter request: ");

    }

    return 0;

}

**-----------------------------------------------------------**

# 

# **Пример работы**

**Test 1**

|  |  |
| --- | --- |
| Program\_second.c Input | Program\_second.c Output |
| 1  5  0  1  5  3 |  |

# **Вывод**