МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский Авиационный Институт

(Национальный исследовательский университет)

Институт №8

«Компьютерные науки и прикладная математика»

Кафедра 806

«Вычислительная математика и программирование»

Курсовой проект по дисциплине «Математический практикум»

Студент: Губарев М. С.

Группа: М8О-211Б-22

Преподаватель: Романенков А. М.

Оценка:

Дата:

Москва 2023

Содержание

Требования к курсовому проекту по Математическому практикуму.

Задание к курсовому проекту:

Необходимо выполнить задания из списков заданий к работам №1, №2, №3, №4.

Каждое задание (1-10) из списков заданий №1, №2, №3 оценивается в 1 балл.

Задания 1, 2 из списка заданий №4 оцениваются в 1 балл.

Задания 3, 4, 5, 6, 7, 8 из списка заданий №4 оцениваются в 2 балла.

Задания 9, 10 из списка заданий №4 оцениваются в 3 балла.

Максимальное количество баллов, которое возможно набрать - 50.

На оценку 3 необходимо набрать 30 баллов.

На оценку 4 необходимо набрать 38 баллов.

На оценку 5 необходимо набрать 45 баллов.

Лабораторная работа № 1.

Задание 1. [1]

Через аргументы командной строки программе подаются число и флаг, определяющий действие с этим числом. Флаг начинается с символа ‘-’ или ‘/’. Программа распознает следующие флаги:

- -h вывести в консоль натуральные числа в пределах 100 включительно, кратные указанному. Если таковых нету – вывести соответствующее сообщение;
- -p определить является ли введенное число простым; является ли оно составным;
- -s разделить число на отдельные цифры и вывести отдельно каждую цифру числа, разделяя их пробелом, от старших разрядов к младшим, без ведущих нулей в строковом представлении;
- -e вывести таблицу степеней (для всех показателей в диапазоне от 1 до заданного числа) оснований от 1 до 10; для этого флага работает ограничение на вводимое число: оно должно быть не больше 10;
- -a вычислить сумму всех натуральных чисел от 1 до указанного числа включительно и вывести полученное значение в консоль;
- -f вычислить факториал указанного числа и вывести полученное значение в консоль.

Решение.

Общая идея решения:

Программа представляет собой консольное приложение, которое принимает через аргументы командной строки число и флаг. Флаг определяет действие, которое программа должна выполнить с этим числом. Возможные флаги включают вывод натуральных чисел, проверку простоты, разделение числа на цифры, вывод таблицы степеней и другие.

Основные функции:

`from_str_to_ll:`

- Функция преобразует строку в беззнаковое длинное целое число (unsigned long long). Она также выполняет проверку на переполнение, наличие букв и дробных частей.

`check_separable_digits_in_range:`

- Функция создает массив разделимых цифр в пределах от 1 до 100 для заданного числа.

`check_for_primary:`

- Функция проверяет, является ли введенное число простым.

`print_primary_check_result:`

- Функция выводит результаты функции `check_for_primary`, сообщая, является ли число простым или составным.

`get_split_by_digits:`

- Функция преобразует строку в массив символов, где каждый символ представляет собой одну цифру.

`validate_input:`

- Функция проверяет, находится ли введенное число в допустимом диапазоне (от 1 до 10) для построения таблицы степеней.

`print_table_of_powers:`

- Функция выводит таблицу степеней для введенного числа в заданном диапазоне, если введенное число допустимо.

`sum_of_numbers:`

- Функция вычисляет сумму натуральных чисел от 1 до введенного числа.

`factorial:`

- Функция вычисляет факториал введенного числа.

`print_factorial:`

- Функция выводит результаты функции factorial, сообщая о факториале, переполнении или недопустимом вводе.

`make_call:`

- Функция осуществляет вызов нужной функции в зависимости от переданного флага.

`validate_input_all:`

- Функция проверяет корректность ввода, удостоверяясь, что передано необходимое количество аргументов командной строки.

Данная программа на языке C представляет собой набор функций, выполняющих различные математические операции и проверки над числами. Она разработана для выполнения различных действий в зависимости от переданного флага в командной строке. Программа предоставляет пользователю возможность выбора различных операций, представляет информацию о числах и их свойствах, а также включает в себя справочную информацию для пользователя.

Задание 2. [2]

Реализуйте функции, вычисляющие значения чисел e, π, $ln\ 2$, $\sqrt{\ }2$, γ с заданной точностью. Для каждой константы реализуйте три способа вычисления: как сумму ряда, как решение специального уравнения, как значение предела.

Замечание. Вы можете использовать следующие факты:

	Предел	Ряд/Произведение	Уравнение
e	$e = \lim\limits_{n\to\infty}\left(1+\frac{1}{n}\right)^n$	$e = \sum\limits_{n=0}^{\infty}\frac{1}{n!}$	$\ln x = 1$
π	$\pi = \lim\limits_{n\to\infty}\frac{(2^n n!)^4}{n((2n)!)^2}$	$\pi = 4\sum\limits_{n=1}^{\infty}\frac{(-1)^{n-1}}{n}$	$\cos x = -1$
$\ln 2$	$\ln 2 = \lim\limits_{n\to\infty} n(2^n - 1)$	$\ln 2 = \sum\limits_{n=1}^{\infty}\frac{(-1)^{n-1}}{n}$	$e^x = 2$
$\sqrt{2}$	$\sqrt{2} = \lim\limits_{n\to\infty} x_n$, где $x_{n+1} = x_n - \frac{x_n^2}{2} + 1,$ $x_0 = -0.5$	$\sqrt{2} = \prod\limits_{k=2}^{\infty} 2^{2^{-k}}$	$x^2 = 2$
γ	$\gamma = \lim\limits_{m\to\infty}\left(\sum\limits_{k=1}^{m} C_m^k \frac{(-1)^k}{k}\ln(k!)\right)$	$\gamma = -\frac{\pi^2}{6} + \sum\limits_{k=2}^{\infty}\left(\frac{1}{\lfloor\sqrt{k}\rfloor^2} - \frac{1}{k}\right)$	$e^{-x} = \lim\limits_{t\to\infty}\left(\ln t \prod\limits_{p\le t, p\in P}\frac{p-1}{p}\right)$

Рисунок 1 математические факты

Точность вычислений (значение эпсилон) подаётся программе в качестве аргумента командной строки.

Решение.

Общая идея решения:

Общая идея решения заключается в создании программы на языке C, которая вычисляет значения математических констант e, π, $ln\ 2$, sqrt(2), и γ

с заданной точностью. Для вычисления каждой константы использованы три способа вычисления: через сумму ряда, решение специального уравнения и вычисление предела.

Вычисление e (число Эйлера):

- `e_limit`: Вычисление числа эйлера через предел.
- `e_tailor`: Вычисление через сумму ряда.
- `e_newton`, `df_e` и `func_e`: Вычисление методом Ньютона.
  Вычисление π (число Пи):
- pi_limit: Вычисление через предел.
- pi_tailor: Вычисление через сумму ряда.
- pi_newton, df_pi и func_pi: Вычисление методом Ньютона.
  Вычисление ln 2 (натуральный логарифм от 2):
- ln2_limit: Вычисление через предел.
- ln2_tailor: Вычисление через сумму ряда.
- ln2_f, ln2_df, ln2_newton: Вычисление методом Ньютона.
  Вычисление √2 (квадратный корень из 2):
- sqrt2_limit: Вычисление через предел.
- sqrt2_tailor: Вычисление через произведение.
- sqrt2_func, sqtr2_df, sqrt2_newton: Вычисление методом Ньютона.
  Вычисление γ (постоянная Эйлера-Маскерони):
- gamma_limit: Вычисление через предел.
- gamma_tailor: Вычисление через сумму ряда.
- gamma_limit_for_eq, gamma_f, gamma_fd, gamma_dirihle: Вычисление методом Дирихле.

Данная программа решает задачу вычисления значений математических констант (e, π, ln 2, √2, γ) с заданной точностью, используя различные методы, такие как сумма ряда, значения предела и вычисления уравнения методом Ньютона или методом Дирихле. Для каждой константы значение вычисляется с помощью трёх вычислительных методов. Программа предоставляет гибкую возможность выбора точности вычислений, предоставляя пользователю управление этим параметром через аргумент командной строки.

Задание 3. [3]

Через аргументы командной строки программе подается флаг, который определяет действие, и набор чисел. Флаг начинается с символа ‘-’ или ‘/’. Необходимо проверять соответствие количества параметров введённому флагу. Программа распознает следующие флаги:

● -q первый параметр (вещественное число) задаёт точность сравнения вещественных чисел (эпсилон), оставшиеся три (вещественные числа) являются коэффициентами квадратного уравнения; необходимо вывести в консоль решения этого уравнения при всевозможных уникальных перестановках значений коэффициентов при степенях переменной;

● -m необходимо задать два ненулевых целых числа, после чего определить, кратно ли первое число второму;

● -t первый параметр (вещественное число) задаёт точность сравнения вещественных чисел (эпсилон); необходимо проверить, могут ли оставшиеся три (вещественные числа) параметра являться длинами сторон прямоугольного треугольника.

Решение.

Общая идея решения:

Программа представляет собой консольное приложение, которое выполняет различные действия в зависимости от переданного через аргумент командной строки флага. Вот общая суть программы и краткое описание ключевых функций:

Основные функции:

`check_fractional:`

- Проверяет корректность дробного числа и возможные проблемы (переполнение, недостаток).

`get_fractional:`

- Преобразует строку в дробное число и проверяет его корректность.

`check_fractional_uzs:`

- Проверяет дробное число на корректность и на возможные проблемы (переполнение, недостаток).

`get_fractional_uzs:`

- Преобразует строку в дробное число поддержки нуля и проверяет его корректность.

`check_for_separable`:

- Проверяет, можно ли разделить первое число на второе без остатка.

`sort_3_ld`:

- Сортирует три значения типа long double в порядке возрастания.

`is_triangle`:

- Проверяет, можно ли построить треугольник с заданными сторонами.

`solve_quadratic`:

- Выводит корни уравнения, если они есть, или сообщение о их отсутствии.

Данные функции позволяют выполнить задачи, описанные в условии: решение квадратного уравнения, проверка кратности чисел, и определение возможности образования прямоугольного треугольника.

Задание 4. [4]

На вход программе, через аргументы командной строки, подается флаг и путь к файлу. Флаг определяет действие с входным файлом. Флаг начинается с символа ‘-’ или ‘/’. Если флаг содержит в качестве второго символа опциональный символ ‘n’ (то есть “-nd”, “/nd”, “-ni”, “/ni”, “-ns”, “/ns”, “-na”, “/na”), то путь к выходному файлу является третьим аргументом командной строки; иначе имя выходного файла генерируется приписыванием к имени входного файла префикса “out_”. Вывод программы должен транслироваться в выходной файл. Программа распознает следующие флаги:

- -d необходимо исключить символы арабских цифр из входного файла;
- -i для каждой строки входного файла в выходной файл необходимо записать сколько раз в этой строке встречаются символы букв латинского алфавита;
- -s для каждой строки входного файла в выходной файл необходимо записать сколько раз в этой строке встречаются символы, отличные от символов букв латинского алфавита, символов арабских цифр и символа пробела;
- -a необходимо заменить символы, отличные от символов цифр, ASCII кодом, записанным в системе счисления с основанием 16.

Решение.

Общая идея решения:

Программа представляет собой консольное приложение, которое выполняет различные действия в зависимости от переданного через аргументы командной строки флага и пути к файлу. Вот общая суть программы и краткое описание ключевых функций:

Основные функции:

`get_fractional_uzs:`

- Преобразует строку в дробное число поддержки нуля и проверяет его корректность.

`try_to_open_w:`

- Пытается открыть файл для записи. Возвращает значение перечисления, указывающее на успешное открытие файла для записи или ошибку.

`create_output_file:`

- Создает путь для нового выходного файла на основе входного пути.

`flag_d:`

- Копирует содержимое входного файла, оставляя только цифры, в выходной файл.

`flag_a:`

- Копирует содержимое входного файла, преобразуя нецифровые символы в шестнадцатеричные, в выходной файл.

`flag_i:`

- Подсчитывает количество латинских букв в каждой строке входного файла и записывает результаты в выходной файл.

`flag_s:`

- Подсчитывает количество неалфавитных и нецифровых символов в каждой строке входного файла и записывает результаты в выходной файл.

Программа решает задачи, связанные с обработкой содержимого файла в зависимости от выбранного флага: исключение арабских цифр, подсчет латинских символов, подсчет символов, отличных от латинских символов, арабских цифр и пробела, а также замена символов их ASCII-кодами в системе счисления с основанием 16.

Задание 5. [5]

Вычислить значения сумм с точностью , где ε (ε вещественное число) подаётся программе в виде аргумента командной строки:

a. $\sum_{n=0}^{\infty} \frac{x^n}{n!}$

b. $\sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}$

c. $\sum_{n=0}^{\infty} \frac{3^{3n}(n!)^3 x^{2n}}{(3n)!}$

d. $\sum_{n=1}^{\infty} \frac{(-1)^n (2n-1)!! x^{2n}}{(2n)!!}$

Рисунок 2 необходимые суммы для вычисления

Решение.

Общая идея решения:

Программа реализует вычисление значений сумм с заданной точностью для четырех различных функций (a, b, c, d) в зависимости от переданных аргументов командной строки. Вот краткое описание ключевых частей программы:

Основные функции:

`char_to_ld:`

- Преобразует строку в формате char в long double.

`check_input:`

- Проверяет корректность ввода данных и инициализирует переменные x и эпсилон.

`sum_a, sum_b, sum_c, sum_d:`

- Реализуют вычисление суммы для четырех различных функций с заданной точностью.

Программа проверяет корректность введенных данных, включая значения эпсилон и флагов. Затем она итеративно вычисляет значения сумм для каждой из четырех функций и выводит результаты в стандартный поток вывода.

Задание 6. [6]

Вычислить значения интегралов с точностью ε, где ε (вещественное число) подаётся программе в виде аргумента командной строки:

a. $\int_0^1 \frac{\ln(1+x)}{x} dx$
b. $\int_0^1 e^{-\frac{x^2}{2}} dx$
c. $\int_0^1 \ln\frac{1}{1-x} dx$
d. $\int_0^1 x^x dx$

Рисунок 3 интегралы для задачи

Решение.

Общая идея решения:

Программа вычисляет значения интегралов с использованием метода Симпсона с заданной точностью эпсилон для четырех различных функций. Вот краткое описание ключевых частей программы:

Основные функции:

`str_to_ld`:
- Преобразует строку в формате char в long double.

`simpsons_method`:
- Вычисляет интеграл методом Симпсона, по указателю на функцию, для которой вычисляется интеграл.

`check_input`:
- Проверяет корректность ввода данных.

`main`:
- Вызывает необходимые функции и выводит их результаты.

`integral_a`, `integral_b`, `integral_c` и `integral_d`:
- Функции вычисляющие интегралы из условия.

Программа проверяет корректность введенных данных, включая значения эпсилон. Затем она вычисляет значения интегралов для каждой из четырех функций, используя метод Симпсона, а затем выводит вычисленные значения, для каждой из функций.

Задание 7. [7]

На вход программе подаётся флаг и пути к файлам. Флаг начинается с символа ‘-’ или “/”. Необходимо проверять соответствие количества параметров введённому флагу.

Программа распознает следующие флаги:

- -r записать в файл, путь к которому подаётся последним аргументом командной строки, лексемы и файлов file1 и file2, пути к которым подаются как третий и четвёртый аргументы командной строки соответственно, где на нечётных позициях находятся лексемы из file1, а на чётных – из file2. Лексемы во входных файлах разделяются произвольным количеством символов пробела, табуляций и переносов строк. Если количество лексем в файлах различается, после достижения конца одного из входных файлов, необходимо переписать в выходной файл оставшиеся лексемы из другого из входных файлов. Лексемы в выходном файле должны быть разделены одним символом пробела.
- -a записать в файл, путь к которому подаётся последним аргументом командной строки, файл, путь к которому подаётся третьим аргументом командной строки, таким образом, чтобы:
  - в каждой десятой лексеме сначала все символы букв латинского алфавита были преобразованы в эквивалентные символы строчных букв латинского алфавита, а затем все символы были преобразованы в эквивалентные им ASCII-коды, записанные в системе счисления с основанием 4;
  - в каждой второй (и одновременно не десятой) лексеме все символы букв латинского алфавита были преобразованы в эквивалентные символы строчных букв латинского алфавита;
  - в каждой пятой (и одновременно не десятой) лексеме все символы были преобразованы в эквивалентные им ASCII-коды, записанные в системе счисления с основанием 8.

Лексемы во входном файле разделяются произвольным количеством символов пробела, табуляций и переносов строк. Лексемы в выходном файле должны быть разделены одним символом пробела.

Решение.

Общая идея решения:

Программа реализует обработку командной строки для выполнения различных операций в зависимости от введенных флагов и файлов. Вот краткое описание ключевых частей программы:

Основные функции:

`skip_empty`:
- Перемещает указатель файла к первому не пустому символу.

`is_latin_letter`:
- Проверяет, является ли символ латинской буквой.

`get_lexema`:
- Извлекает лексему из файла. Возвращает указатель на динамически выделенную строку (лексему) или NULL в случае ошибки.

`read_and_write_r`:
- Читает из двух файлов и записывает в третий файл, разделяя лексемы пробелами.

`write_to_file_in_cc`:
- Записывает числовое значение в файл, используя систему счисления CC.

`to_ascii_and_cc`:
- Преобразует символы лексемы в ASCII и записывает их в файл, используя систему счисления CC.

`write_to_file_a`:
- Записывает лексемы в файл, применяя различные преобразования к некоторым из них.

`check_flag`:
- Проверяет корректность флага в командной строке.

`check_input_validate_n_run`:
- Проверяет входные параметры и запускает соответствующую операцию.

Программа обрабатывает различные сценарии ввода и выводит соответствующие сообщения об ошибках или успешном выполнении. Обработка различных сценариев ошибок и информационных запросов делает программу более удобной для использования и понимания пользователем.

Задание 8. [8]

В текстовом файле, путь к которому подаётся как второй аргумент командной строки, находятся числа записанные в разных системах счисления, при этом информация о конкретной системе счисления для каждого числа утеряна. В файле числа разделены произвольным количеством разделителей (символов пробела, табуляций и переносов строки). Для каждого числа из входного файла программа должна определить минимальное основание системы счисления (в диапазоне [2..36]), в которой представление этого числа корректно, и в выходной файл, путь к которому подаётся третьим аргументом командной строки, построчно выводит: входное число без ведущих нулей, определенное для него минимальное основание системы счисления и представление этого числа в системе счисления с основанием 10. Прописные и строчные символы букв латинского алфавита отождествляются.

Решение.

Общая идея задачи:

Программа анализирует числа в разных системах счисления из текстового файла и выводит результаты в другой файл. Вот краткое описание ключевых частей программы:

Основные функции:

`char_digit_to_number(char digit):`

- Преобразует символ-цифру в числовое значение в соответствии с системой счисления.

`ss_to_base_10:`

- Преобразует строку числа в указанной системе счисления в десятичное число.

`find_min_base:`

- Определяет минимальную основание системы счисления, которое может представить символ.

`check_input:`

- Проверяет корректность входных параметров, открывает входной и выходной файлы.

`skip_empty:`

- Перемещает указатель файла к первому не пустому символу.

`get_lexema:`

- Извлекает лексему из файла, определяет минимальное основание системы счисления.

`cycle:`

- Обрабатывает входной файл, преобразуя числа из разных систем счисления в десятичную систему и выводит результаты в выходной файл.

Программа выполняет обработку файла, содержащего числа, и определяет минимальное основание системы счисления для каждого числа. Результаты обработки выводятся в указанный файл. Программа также обрабатывает ошибки, такие как переполнение и некорректные числа.

Задание 9. [9]

1. Заполнить массив фиксированного размера псевдослучайными числами в диапазоне [a..b], где a, b задаются в качестве аргументов командной строки. Реализовать функцию, выполняющую поиск максимального и минимального значений элементов массива и меняющую местами максимальный и минимальный элементы в исходном массиве за один проход по нему.
2. Заполнить динамические массивы A и B псевдослучайного размера в диапазоне [10..10000] псевдослучайными числами в диапазоне [-1000..1000]. Сформировать из них динамический массив C, где i-й элемент массива C есть i-й элемент массива A, к которому добавлено значение ближайшего к A[i] по значению элемента из массива B.

Решение.

Основные функции:

`swap_maxmin_elements`:

- Функция, выполняющая поиск максимального и минимального значений в массиве и меняющая их местами.

`is_number:`

- Проверяет, является ли строка числом.

`check_input:`

- Проверяет корректность входных параметров.

`swap:`

- Обменивает значения между двумя переменными.

`get_random_value:`

- Генерирует случайное значение в заданном диапазоне [a, b].

`fill_array_rand:`

- Заполняет массив случайными значениями в заданном диапазоне.

`print_mass:`

- Выводит массив на экран.

`swap_max_min_elements:`

- Обменивает максимальный и минимальный элементы массива.

`is_number:`

- Проверяет, является ли строка числом.

`get_nearest_bs:`

- Находит ближайший элемент к x в упорядоченном массиве mass с использованием бинарного поиска.

`generate_c:`

- Генерирует массив c, суммируя элементы массива a с их ближайшими элементами из массива b.

Программа выполняет обработку массивов содержащих числа, и выполняющая над ними действия. Программа также обрабатывает ошибки, такие как переполнение и некорректные числа.

Задание 10. [10]

Пользователь вводит в консоль основание системы счисления (в диапазоне [2..36]) и затем строковые представления целых чисел в системе счисления с введённым основанием. Окончанием ввода является ввод строки “Stop”. Найти среди введённых чисел максимальное по модулю. Напечатать его без ведущих нулей, а также его строковые представления в системах счисления с основаниями 9, 18, 27 и 36.

Решение.

Общая идея решения:

Программа предоставляет пользователю возможность ввода чисел в различных системах счисления с указанным основанием. После ввода чисел и строки "Stop" программа определяет число с максимальным модулем, выводит его без ведущих нулей и представляет в системах счисления с основаниями 9, 18, 27 и 36.

Основные функции и структуры:

`enum input_statements:`

- Перечисление состояний ввода, используемых для обозначения различных сценариев обработки ввода.

`check_base:`

- Проверяет корректность значения основания системы счисления.

`char_digit_to_number:`

- Преобразует символ-цифру в числовое значение в соответствии с системой счисления.

`ss_to_base_10:`

- Преобразует строку числа в указанной системе счисления в десятичное число.

`get_len:`

- Определяет длину числа в новой системе счисления.

`ll_to_cc:`

- Преобразует десятичное число в указанной системе счисления в строку.

Программа обрабатывает ввод строки "Stop" для завершения ввода чисел. В случае возникновения ошибок, таких как неверное основание

системы счисления или некорректные числа, программа выводит соответствующие сообщения.

Лабораторная работа № 2.

Задание 1. [11]

Через аргументы командной строки программе подаются флаг (первым аргументом), строка (вторым аргументом); и (только для флага -c) целое число типа unsigned int и далее произвольное количество строк. Флаг определяет действие, которое необходимо выполнить над переданными строками:

● -l подсчёт длины переданной строки, переданной вторым аргументом;

● -r получить новую строку, являющуюся перевёрнутой (reversed) переданной вторым аргументом строкой;

● -u получить новую строку, идентичную переданной вторым аргументом, при этом каждый символ, стоящий на нечетной позиции (первый символ строки находится на позиции 0), должен быть преобразован в верхний регистр;

● -n получить из символов переданной вторым аргументом строки новую строку так, чтобы в начале строки находились символы цифр в исходном порядке, затем символы букв в исходном порядке, а в самом конце – все остальные символы, также в исходном порядке;

● -c получить новую строку, являющуюся конкатенацией второй, четвертой, пятой и т. д. переданных в командную строку строк; строки конкатенируются в псевдослучайном порядке; для засеивания генератора псевдослучайных чисел функцией srand используйте seed равный числу, переданному в командную строку третьим аргументом.

Для каждого флага необходимо реализовать соответствующую ему собственную функцию, выполняющую действие. Созданные функциями строки должны располагаться в выделенной из динамической кучи памяти. При реализации функций запрещено использование функций из заголовочного файла string.h. Продемонстрируйте работу реализованных функций.

Решение.

Общая идея решения:

Анализ входного флага и проверка правильности переданных аргументов осуществляются функцией-парсером входных данных. После определения

входного флага, к строке можно применять ряд функций, избегая использование библиотеки <string.h>:

Основные функции:

- `is_lower_altha_s`: Проверяет, является ли символ строчной латинской буквой.
- `is_altha_s`: Проверяет, является ли символ латинской буквой.
- `is_digit_s`: Проверяет, является ли символ цифрой.
- `to_upper`: Возвращает символ в верхнем регистре, если он является строчной латинской буквой.
- `strlen_s`: Возвращает длину строки.
- `get_reversed`: Возвращает обратную строку.
- `operator_u`: Применяет операцию "u" к строке.
- `operator_n`: Применяет операцию "n" к строке.
- `swap_c`: Обменивает значениями две строки.
- `operator_c`: Применяет операцию "c" к массиву строк.
- `strcmp_s`: Сравнивает две строки.
- `check_input`: Проверяет валидность входных параметров.
- `err_mess`: Выводит сообщение об ошибке в зависимости от кода ошибки.

В программе были реализованы необходимые функции, а также обработаны случаи, где возможно возникновение ошибок.

Задание 2. [12]

1. Реализуйте функцию с переменным числом аргументов, вычисляющую среднее геометрическое переданных ей чисел вещественного типа. Количество (значение типа int) переданных вещественных чисел задается в качестве последнего обязательного параметра функции.

2. Реализуйте рекурсивную функцию возведения вещественного числа в целую степень. При реализации используйте алгоритм быстрого возведения в степень. Продемонстрируйте работу реализованных функций

Решение.

Общая идея решения:

Вычисление среднего геометрического для введенных значений осуществляется путем последовательного умножения всех введенных значений друг на друга, а затем извлечения корня n-ой степени, где n - количество введенных аргументов. Бинарное возведение в степень достигается сохранением предыдущих результатов и вследствии уменьшении числа операций умножения.

Основные функции:

- `geometric_middle`: Вычисляет среднее геометрическое переданных аргументов. Аргумент num указывает количество аргументов, за которыми следуют числа переменной длины.
- `__b_pow_d`: Вспомогательная рекурсивная функция для вычисления степени числа типа long double.
- `b_pow_d`: Вычисляет степень числа типа long double. Поддерживает отрицательные степени.

В программе были реализованы необходимые функции, а также обработаны случаи, где возможно возникновение ошибок.

Задание 3. [13]

Реализуйте функцию с переменным числом аргументов, принимающую в качестве входных параметров подстроку и пути к файлам. Необходимо чтобы для каждого файла функция производила поиск всех вхождений переданной подстроки в содержимом этого файла. При реализации запрещается использование функций strstr и strncmp из заголовочного файла string.h. Продемонстрируйте работу функции, также организуйте наглядный вывод результатов работы функции: для каждого файла, путь к которому передан как параметр Вашей функции, для каждого вхождения подстроки в содержимое файла необходимо вывести номер строки (индексируется с 1) и номер символа в строке файла, начиная с которого найдено вхождение подстроки. В подстроку могут входить любые символы (обратите внимание, что в подстроку также могут входить символы пробела, табуляций, переноса строки, в неограниченном количестве)

Решение.

Общая идея решения:

Программа решает задачу путем просмотра файла и поиска наибольшей совпадающей подстроки символов в файле.

Основные структуры и функции:

- `is_equal_str`: Сравнивает две строки до заданной длины.
- `move_char_left`: Сдвигает символы строки влево.
- `move_int_left`: Сдвигает элементы массива целых чисел влево.
- `Cell`: Структура, представляющая результат поиска паттерна. Содержит информацию о найденных строках и символах.
- `find_pattern`: Ищет паттерн в файле. Возвращает структуру `Cell` с результатами.
- `free_Cell`: Освобождает память, занятую структурой `Cell`.
- `free_Cells`: Освобождает память, занятую массивом структур `Cell`.
- `find_all_patterns`: Ищет заданный паттерн в нескольких файлах. Возвращает массив структур `Cell` с результатами.

Решение сводится к прохождению по файлу и поиску максимально

длинного совпадения одинаковых символов подстроки и символов в файле.

Задание 4. [14]

1. Реализуйте функцию с переменным числом аргументов, принимающую координаты (вещественного типа, пространство двумерное) вершин многоугольника и определяющую, является ли этот многоугольник выпуклым.

2. Реализуйте функцию с переменным числом аргументов, находящую значение многочлена степени n в заданной точке. Входными параметрами являются точка (вещественного типа), в которой определяется значение многочлена, степень многочлена (целочисленного типа), и его коэффициенты (вещественного типа, от старшей степени до свободного коэффициента в порядке передачи параметров функции слева направо). Продемонстрируйте работу реализованных функций

Решение.

Общая идея решения:

Программа обрабатывает координаты вершин многоугольника, использует метод правых троек для выдачи ответа. Для вычисления значения полинома применяется схема Горнера.

Основные функции:

- `isConvexPolygon`: Проверяет, является ли многоугольник выпуклым. Принимает количество точек и координаты точек в формате (x1, y1, x2, y2, ..., xn, yn).
- `polynomialValue`: Вычисляет значение полинома для заданного значения x. Принимает количество коэффициентов и сами коэффициенты в формате (a0, a1, a2, ..., an).

В программе были реализованы необходимые функции, а также обработаны случаи, где возможно возникновение ошибок.

Задание 5. [15]

Реализуйте функции overfprintf и oversprintf, поведение которых схоже с поведением стандартных функций fprintf и sprintf, то есть эти функции имеют одинаковый прототип и логику работы, но в ваших функциях помимо стандартных флагов определены следующим образом дополнительные флаги:

- %Ro – печать int, записанного римскими цифрами;
- %Zr – печать в поток вывода цекендорфова представления целого числа
- %Cv печать целого числа типа int в системе счисления с заданным основанием
- %CV – аналогично флагу %Cv, но в верхнем регистре;
- %to – результат перевода целого числа, записанного в строковом представлении в системе счисления с заданным основанием в систему счисления с основанием 10
- %TO - аналогично флагу %to, но верхний регистр
- %mi – печать дампа памяти для типа int
- %mu – печать дампа памяти для типа unsigned int
- %md – печать дампа памяти для типа double
- %mf – печать дампа памяти для типа float

Решение.

Общая идея решения:

Программа является расширенной версией функции форматирования строк, которые не поддерживаются стандартной библиотекой C.

Основные функции:

- `validate_int`: Валидирует целое число в заданном диапазоне.
- `to_roman`: Преобразует целое число в римскую систему счисления.
- `get_fib`: Возвращает массив чисел Фибоначчи не превышающих заданный лимит.
- `zeckendorf_look`: Преобразует число в представление числа Цекендорфа .
- `_10th_to_cc`: Преобразует целое число в систему счисления с произвольным основанием.
- `cc_to_10th`: Преобразует число в системе счисления с произвольным основанием в десятичную систему..

- `print_memory_dump`: Выводит представление памяти в виде битовой последовательности.
- `split_perc`: Разбивает строку по символу '%' и возвращает массив из трех подстрок.
- `oversprintf`: Функция, аналогичная sprintf, но с поддержкой дополнительных форматов.
- `overfprintf`: Функция, аналогичная fprintf, но с поддержкой дополнительных форматов.

В программе были реализованы необходимые функции, а также обработаны случаи, где возможно возникновение ошибок.

Задание 6. [16]

Реализуйте функции overfscanf и oversscanf, поведение которых схоже с поведением стандартных функций fscanf и sscanf соответственно, то есть эти функции имеют одинаковый прототип и логику работы, но в ваших функциях помимо стандартных флагов добавляются дополнительные флаги:

- %Ro – считывание из потока ввода целого числа типа int, записанного римскими цифрами;
- %Zr – считывание из потока ввода целого числа типа unsigned int, записанного в виде цекендорфова представления
- %Cv считывание из потока ввода целого числа типа int, записанного в системе счисления с заданным основанием
- %CV – аналогично флагу %Cv, но в верхнем регистре

Решение.

Общая идея решения:

В целом, программа представляет собой расширенную версию функции считывания строк, которая может обрабатывать специфические задачи преобразования данных, не предусмотренные стандартной библиотекой C.

Основные функции:

- `split_perc`: Функция разбивает строку по символу '%' и возвращает массив из трех подстрок. Первая подстрока до первого '%', вторая - между первым '%' и следующим пробелом, третья - всё остальное.
- `move`: Выполняет перемещение указателя в зависимости от типа данных.
- `overfscanf`: Функция, аналогичная fscanf, но с поддержкой дополнительных форматов, таких как %Ro, %Zr, %Cv, %CV.
- `oversscanf`: Функция, аналогичная sscanf, но с поддержкой дополнительных форматов, таких как %Ro, %Zr, %Cv, %CV.
- `from_roman`: Преобразует строку с римскими цифрами в целое число. Результат сохраняется в переменной, на которую указывает параметр result.
- `zeckendorf_look_int`: Преобразует строку в представлении числа

Цекендорф. Результат сохраняется в переменной, на которую указывает параметр res.

- `ss_to_base_10`: Преобразует строку, представляющую число в определенной системе счисления, в целое число. Основание системы счисления может быть в диапазоне от 2 до 36. Результат сохраняется в переменной, на которую указывает параметр result.

В программе были реализованы необходимые функции, а также обработаны случаи, где возможно возникновение ошибок.

Задание 7. [17]

Реализуйте функцию, которая находит корень уравнения одной переменной методом дихотомии. Аргументами функции являются границы интервала, на котором находится корень; точность (эпсилон), с которой корень необходимо найти, а также указатель на функцию, связанной с уравнением от одной переменной. Продемонстрируйте работу функции на разных значениях интервалов и точности, для различных уравнений.

Решение.

Общая идея решения:

Функция dichotomy реализует метод дихотомии для поиска корней уравнения в интервале [a, b] с использованием указанной функции function и заданной точности eps. Метод дихотомии разделяет интервал пополам и выбирает тот подинтервал, на котором функция меняет знак. Процесс повторяется, пока длина интервала не станет меньше eps.

Основные функции:

- `ex_1, ex_2, ex_3`: Предлагаемые уравнения.
- `dichotomy`: Функция метода дихотомии.

Таким образом, эта программа вызывает функцию `dichotomy` для нахождения корней функций `ex_1, ex_2, ex_3,` с заданной точностью. Результаты находятся методом дихотомии и выводятся в стандартный поток вывода.

Задание 8. [18]

Реализуйте функцию с переменным числом аргументов, вычисляющую сумму переданных целых неотрицательных чисел в заданной системе счисления с основанием [2..36]. Параметрами функции являются основание системы счисления, в которой производится суммирование, количество переданных чисел, строковые представления чисел в заданной системе счисления. Десятичное представление переданных чисел может быть слишком велико и не поместиться во встроенные типы данных; для решения возникшей проблемы также реализуйте функцию «сложения в столбик» двух чисел в заданной системе счисления, для её использования при реализации основной функции. Результирующее число не должно содержать ведущих нулей.

Решение.

Общая идея решения:

Данный код решает задачу сложения чисел произвольной длины в произвольной системе счисления.

Основные функции:

- `format_num`: Убирает ведущие нули из строки числа.
- `move_right_char`: Сдвигает символы строки вправо.
- `get_num`: Возвращает числовое представление символа (цифры) в соответствии с текущей системой счисления.
- `max`: Возвращает максимальное из двух чисел.
- `sum_num`: Складывает два числа в указанной системе счисления и возвращает результат в виде строки. При необходимости расширяет выделенную под результат память.
- `sum_nums`: Складывает произвольное количество чисел в указанной системе счисления и возвращает результат в виде строки.

Таким образом, программа позволяет складывать произвольное количество чисел в разных системах счисления.

Задание 9. [19]

Реализуйте функцию с переменным числом аргументов, определяющую для каждой из переданных десятичных дробей (в виде значения вещественного типа в диапазоне (0; 1)), имеет ли она в системе счисления с переданным как параметр функции основанием конечное представление. Продемонстрируйте работу функции.

Решение.

Общая идея решения:

Задача сводится к определению подмножества простых делителей знаменателя дроби и сравнение их с простыми делителями целевой системы счисления.

Основные функции:

- `has_finite_representation`: Проверяет, имеет ли число конечное представление в заданной системе счисления. Для этого число умножается на основание системы счисления до тех пор, пока не станет целым. Затем проверяется, что числитель и знаменатель не имеют общих делителей, кроме 1.
- `check_representation`: Принимает переменное количество чисел с плавающей запятой и проверяет их представление в заданной системе счисления. Возвращает массив булевых значений, где каждый элемент указывает, имеет ли соответствующее число конечное представление.

В общем, программа осуществляет несколько операций: разложение числа на простые множители, вычисление знаменателя для представления десятичной дроби с фиксированной точностью, а также сравнение представлений десятичных дробей в различных системах счисления путем сравнения массивов делителей.

Задание 10. [20]

Реализуйте функцию с переменным числом аргументов, находящую переразложение многочлена со степенями значения x по степеням значения (x - a). Входными аргументами этой функции являются точность ε, значение a (вещественного типа), указатель на место в памяти, по которому должен быть записан указатель на динамически выделенную в функции коллекцию коэффициентов результирующего многочлена, степень многочлена (целочисленного типа) и его коэффициенты (вещественного типа), передаваемые как список аргументов переменной длины. То есть, если задан многочлен $f(x)$, то необходимо найти коэффициенты g_0, g_1 , …, g_n такие что:

$$f(x) = f_0 + f_1x + f_2x^2 + \cdots + f_nx^n = g_0 + g_1(x-a) + g_2(x-a)^2 + \cdots + g_n(x-a)^n$$

Решение.

Общая идея решения:

Основная идея решения заключается в том, что для разложения многочлена на другие коэффициенты необходимо вычислить производные от текущего многочлена.

Основные функции:

- `calculate`: Вычисляет значение многочлена в точке x для заданных коэффициентов mas и степени n.
- `differ`: Производит операцию дифференцирования многочлена, уменьшая его степень и вычисляя новые коэффициенты.
- `compute_coefficients`: Принимает переменное количество коэффициентов и вычисляет коэффициенты многочлена Лагранжа для данных коэффициентов. Результатом является массив коэффициентов.

Следовательно, приложение создает многочлен, вычисляет его производные, выводит коэффициенты на экран и проверяет значения многочлена в заданных точках.

Лабораторная работа № 3.

Задание 1. [21]

Реализуйте функцию перевода числа из десятичной системы счисления в систему счисления с основанием 2r, r = 1,...,5. При реализации функции разрешается использовать битовые операции и операции обращения к памяти, запрещается использовать стандартные арифметические операции. Продемонстрируйте работу реализованной функции.

Решение.

Общая идея решения:

Программа преобразует целое число из десятичной системы счисления в систему счисления с заданным основанием. Данная реализация использует битовые операции и операции обращения к памяти, а стандартные арифметические операции запрещены. Вот краткое описание ключевых частей программы [31]:

Основные функции:

`plus:`

- с помощью битовых операций суммирует два числа.

`negative :`

- делает число отрицательным.

`reverse_string:`

- разворачивает строку.

`prepare:`

- производит валидации, и получает вспомогательную маску для числа.

`number_to_cc:`

выполняет перевод числа в нужную систему счисления.

Данная программа реализует преобразование целых чисел из десятичной системы, в указанную пользователем в параметрах функции, через бинарные операции XOR, AND, NOT и битовых сдвигов. В коде также реализована обработка ошибок и очистка выделенной памяти.

Задание 2. [22]

Напишите функцию с переменным числом аргументов, на вход которой передаются: размерность пространства n; экземпляры структур, содержащие в себе координаты векторов из n-мерного пространства; указатели на функции, вычисляющие нормы векторов. Ваша функция должна для каждой переданной нормы вернуть самый длинный переданный вектор

Замечание. Реализуйте возможность вычислять следующие нормы:

$$||x||_\infty = max_j|x_j|$$

$$||x||_p = \left(\sum_{j=1}^{n}|x_j|^p\right)^{\frac{1}{p}}, p \geq 1$$

$$||x||_a = \sqrt{(Ax, x)}$$

Рисунок 4 нормы для задачи

где A - положительно определенная матрица размерности $n \times n$. Передачу функций реализуйте с помощью универсального типа указателя на функцию. Продемонстрируйте работу реализованной функции.

Решение.

Общая идея решения:

Программа находит самые длинные векторы для каждой переданной нормы. Программа демонстрирует использование указателей на функции и переменное число аргументов.

Основные структуры и функции:

`task:`

- Основная функция, вычисляющая для каждого вектора, все переданные нормы, и, возвращающая двумерный массив Vector (содержит самые длинные вектора для каждой из норм).

`print_vector:`

- вывод Vector в стандартный поток вывода.

`get_rand_vector:`

- Возвращает вектор длины n, заполненный случайными значениями.

`Vector:`

- Структура вектор, имеет координаты и размерность.

`norm_pointer:`

- Указатель на функцию вычисляющую норму вектора.
- `statements:`

перечисление описывающее типы ошибок, для их последующей обработки.

Представленный код на языке C представляет собой программу, включающую в себя структуры и функции для работы с векторами в n-мерном пространстве. В ней используется указатель на функцию и функция с переменным количеством аргументов для нахождения самых длинных векторов с использованием различных норм (бесконечной, p-нормы и нормы, определенной матрицей A). Кроме того, в коде реализованы механизмы обработки ошибок и освобождения выделенной памяти.

Задание 3. [23]

На вход программе через аргументы командной строки подается путь ко входному файлу, флаг (флаг начинается с символа ‘-’ или ‘/’, второй символ - ‘a’ или ‘d’) и путь к выходному файлу. В файле в каждой строчке содержится информация о сотруднике (для этой информации определите тип структуры Employee): id (целое неотрицательное число), имя (непустая строка только из букв латинского алфавита), фамилия (непустая строка только из букв латинского алфавита), заработная плата (неотрицательное вещественное число). Программа должна считать записи из файла в динамический массив структур и в выходной файл вывести данные, отсортированные (с флагом ‘-a’/‘/a’ - по возрастанию, с флагом ‘-d’/‘/d’ - по убыванию) первично - по зарплате, далее (если зарплаты равны) - по фамилии, далее (если зарплаты и фамилии равны) - по именам, наконец, по id. Для сортировки коллекции экземпляров структур используйте стандартную функцию qsort, своя реализация каких-либо алгоритмов сортировки не допускается

Решение.

Общая идея решения:

Программа осуществляет чтение данных о сотрудниках из файла, проводит их сортировку сначала по заработной плате, затем по фамилии, имени и id, и затем выводит отсортированные данные в выходной файл в соответствии с указанным флагом ('a' или 'd').

Основные функции:

- `comp_right` и `comp_left`:
- Функции сравнения для использования в функции qsort. Они сравнивают элементы структуры Employee по зарплате, фамилии, имени и id.

`validate_params`:

- функция осуществляющая валидацию входных данных.

`comp_right и comp_left`:

- функции компараторы, для последующего использования в qsort.

`print_employers`:

- вывод информации о работниках.

`scan_eployee`:

- чтение информации о работнике из файла.

Использованные структуры и перечисления:

`Employee:`

- структура для информации о сотруднике, с полями по заданию.

`statements:`

- Перечисление содержащее типы ошибок, для их обработки.

Данная программа на языке C предназначена для считывания информации о сотрудниках из файла, их сортировки, а затем вывода результатов в другой файл. В процессе выполнения используются динамические массивы и стандартная функция qsort для эффективной сортировки данных в заданном порядке. Также предусмотрена обработка ошибок и контроль памяти, что обеспечивает стабильность и надежность программы.

Задание 4. [24]

1. Опишите тип структуры String, содержащую в себе поля для указателя на динамический массив символов типа char и количества символов (длины строки) типа int.

Для описанного типа структуры реализуйте функции:

- создания экземпляра типа String на основе значения типа char *
- удаления внутреннего содержимого экземпляра типа String
- отношения порядка между двумя экземплярами типа String (первично по длине строки, вторично по лексикографическому компаратору)
- отношения эквивалентности между двумя экземплярами типа String (лексикографический компаратор)
- копирования содержимого экземпляра типа String в существующий экземпляр типа String
- копирования содержимого экземпляра типа String в новый экземпляр типа String, размещённый в динамической памяти
- конкатенации к содержимому первого экземпляра типа String содержимого второго экземпляра типа String.

2. Экземпляр структуры Mail содержит в себе экземпляр структуры Address получателя (город (непустая строка), улица (непустая строка), номер дома (натуральное число), корпус (строка), номер квартиры (натуральное число), индекс получателя (строка из шести символов цифр)), вес посылки (неотрицательное вещественное число), почтовый идентификатор (строка из 14 символов цифр), время создания (строка в формате “dd:MM:yyyy hh:mm:ss”), время вручения (строка в формате “dd:MM:yyyy hh:mm:ss”). Экземпляр структуры Post содержит указатель на экземпляр структуры Address текущего почтового отделения и динамический массив экземпляров структур типа Mail. Реализуйте интерактивный диалог с пользователем, предоставляющий функционал для добавления и удаления объектов структур типа Mail в объект структуры типа Post, информативный вывод данных об отправлении при поиске объекта типа Mail по идентификатору. Объекты структуры Mail должны быть отсортированы по индексу получателя (первично) и идентификатору посылки (вторично) в произвольный момент времени. Также в

интерактивном диалоге реализуйте опции поиска всех доставленных отправлений, а также всех отправлений, срок доставки которых на текущий момент времени (системное время) истёк. Информацию о доставленных/недоставленных отправлениях выводите в порядке времени создания по возрастанию (от старых к новым). Для хранения строковых данных используйте структуру String из п. 1.

Решение.

Общая идея решения:

Программа реализует систему управления почтовыми отправлениями, используя структуры данных для представления информации о посылках и почтовых отделениях. Взаимодействие с пользователем осуществляется через интерактивный диалог в консоли.

Основные функции:

`delete_adress:`
- Освобождает память, выделенную под структуру Adress.

`delete_mail:`
- Освобождает память, выделенную под структуру Mail.

`generate_post_id:`
- Генерирует случайный идентификатор почты.

`print_adress, print_mail, print_mails:`
- Выводят информацию об адресе и почтовых отправлениях.

`get_current_time:`
- Возвращает текущее время в нужном формате.

`parse_timestamp:`
- Преобразует строку времени в структуру tm.

`add_mail:`
- Добавляет новое письмо в почтовое отделение.

`search_mail, search_mails:`
- Ищут письмо по идентификатору или отсортированные списки доставленных и просроченных.

`get_b_line, get_int, get_double:`
- Считывают строку, целое и вещественное число соответственно.

`skip_empty:`
- Пропускает пустые символы.

`track_mail:`
- Позволяет отслеживать и изменять статус почтового отправления.

Использованные структуры и перечисления:

`Adress:`

- Структура для информации об адресе

`Mail:`

- Структура для информации о почте, включающая в себя поле адреса.

`Post`:

- Структура для информации о почтовом отделении.

`String`:

- Структура для хранения строк.

Данное программное решение представляет систему управления почтовыми отправлениями. Она использует структуры для хранения данных о получателях, почтовых отправлениях и почтовых отделениях. Пользователь взаимодействует с программой через интерактивный диалог, позволяющий добавлять, искать и удалять почтовые отправления. Программа также предоставляет информацию о доставленных и просроченных отправлениях. Для повышения удобства ввода данных и обеспечения стабильной работы, в программе используется динамическое выделение памяти и обработка ошибок.

Задание 5. [25]

Экземпляр структуры типа Student содержит поля: id студента (целое неотрицательное число), имя (непустая строка только из букв латинского алфавита), фамилия (непустая строка только из букв латинского алфавита), группа (непустая строка) и оценки за 5 экзаменов (динамический массив элементов типа unsigned char). Через аргументы командной строки программе на вход подаётся путь к файлу, содержащему записи о студентах. При старте программа считывает поданный файл в динамический массив структур типа Student. В программе должен быть реализован поиск всех студентов по:

- id;
- фамилии;
- имени;
- группе,

Сортировка (для сортировки необходимо передавать компаратор для объектов структур) студента(-ов) по:

- id;
- фамилии;
- имени;
- группе.

Добавьте возможность вывода в трассировочный файл (путь к файлу передаётся как аргумент командной строки) вывести данные найденного по id студента: ФИО, группу и среднюю оценку за экзамены. Также добавьте возможность вывести в трассировочный файл фамилии и имена студентов, чей средний балл за все экзамены выше среднего балла за все экзамены по всем считанным из файла студентам. Все вышеописанные опции должны быть выполнимы из контекста интерактивного диалога с пользователем.Для сортировки коллекции экземпляров структур используйте стандартную функцию qsort, своя реализация каких-либо алгоритмов сортировки не допускается.

Решение.

Общая идея решения:

Программа представляет собой систему обработки данных о студентах. Она осуществляет чтение информации о студентах из файла, предоставленного в виде аргумента командной строки, и предоставляет пользователю интерактивный интерфейс для выполнения различных действий, таких как поиск, сортировка и запись информации в файл.

Основные функции:

`update_data:`
- обновляет данные в массиве типа Student.

`delete_student:`
- Освобождает память, выделенную для структуры Student.

`create_student:`
- Выделяет память под новую структуру Student и инициализирует её поля.

`create_student_s:`
- Создает структуру Student с использованием строковых данных для имени, фамилии и группы.

`sort_students:`
- Сортировка массива студентов с использованием заданной функции сравнения.

`calculate_mid:`
- Вычисление средней оценки студентов в массиве.

`get_student_info:`
- Создание строки с отформатированной информацией о студенте.

`get_better_mid_students:`
- Выбор студентов с оценками выше средней.

`print_student:`
- Вывод информации о студенте в файл.

`trace_student:`
- Вывод подробной информации о студенте в файл.

`print_students:`
- Вывод информации о массиве студентов в файл.

`trace_students:`
- Запись подробной информации о массиве студентов в файл.

`read_student:`
- Считывание информации о студенте из файла и создание структуры Student.

Использованные структуры и перечисления:

`Student:`

- структура описывающая студента, согласно заданию.

`statements:`

- перечисление описывающее типы ошибок, для их последующей обработки.

Программа обеспечивает обработку ошибок при вводе данных, выделении памяти, открытии файлов и других операциях, предоставляя соответствующие сообщения пользователю для удобства восприятия.

Задание 6. [26]

В некотором уездном городе ввели систему учета городского транспорта. Каждый остановочный пункт регистрирует каждую единицу останавливающегося общественного транспорта, то есть в текстовый файл записывается номер (непустая строка) транспортного средства, время остановки (строка в формате “dd.MM.yyyy hh:mm:ss”), время отправления (строка в формате “dd.MM.yyyy hh:mm:ss”) и маркер, указывающий является ли данная остановка промежуточной, начальной или конечной для данного транспортного средства. Каждый файл содержит координаты остановочного пункта, на котором этот файл был сгенерирован. Необходимо разработать приложение для обработки данных с остановочных пунктов. Реализуйте на базе односвязного списка односвязных списков хранилище информации. Элементами основного списка являются списки пройденных маршрутов от начальной до конечной точки со всеми промежуточными точками для каждого транспортного средства. Элементы в списке пройденных остановок необходимо упорядочить по возрастанию временных меток. Входными аргументами вашего приложения является массив путей к файлам (с каждого остановочного пункта), при этом файлы подаются в произвольном порядке, записи в файле также не упорядочены не по какому-либо критерию, но гарантия целостности записей поддерживается. Для вашего хранилища маршрутов посредством интерактивного диалога реализуйте поиск транспортного средства, которое проехало больше/меньше всех маршрутов, поиск транспортного средства, которое проехало самый длинный/короткий путь, поиск транспортного средства с самым длинным/коротким маршрутом и транспортное средство с самой долгой/короткой остановкой на остановочном пункте и транспортное средство с самым большим временем простоя (стоянки на своих остановках).

Решение

Общая идея решения:

Программа решает задачу обработки информации о движении автобусов. Она читает данные из файла, представляющего собой информацию о координатах автобусов, их маршрутах и времени движения. Затем она выполняет различные анализы и вычисления на основе этих данных.

Основные функции:

`clear_buffer:`
- Очищает буфер ввода файла.

`print_in_list:`
- Выводит информацию о движении автобуса из списка маршрутов.

`add_out:`
- Добавляет информацию о движении автобуса в список всех автобусов.

`parsing_input_data:`
- Считывает и обрабатывает входные данные из файла, добавляя информацию о движении автобусов.

`free_list_buses:`
- Освобождает память, выделенную для списка всех автобусов.

`check_coordination:`
- Проверяет наличие одинаковых координат во всех маршрутах всех автобусов.

`check_time_bus:`
- Проверяет корректность времени во всех маршрутах всех автобусов.

`check_in_list:`
- Проверяет правильность структуры маршрута в списке.

`check_all:`
- Проверяет правильность структуры маршрутов во всех списках автобусов.

`bus_id_length_track:`
- Возвращает номер автобуса с максимальной или минимальной длиной маршрута.

`longest_route:`
- Возвращает максимальную или минимальную длину маршрута в списке.

`max_route_length_bus_id:`
- Возвращает номер автобуса с максимальной или минимальной длиной маршрута в списке всех автобусов.

`max_min_wait_time:`
- Возвращает номер автобуса с максимальным или минимальным временем простоя.

`show_menu:`
- Выводит интерактивное меню для взаимодействия с пользователем и выбора анализа данных.

Каждая из представленных функций выполняет конкретную задачу, необходимую для анализа данных о движении автобусов. Код охватывает разнообразные проверки и вычисления, делая его ценным инструментом для анализа и оптимизации системы маршрутов автобусов.

Задание 7. [27]

В текстовом файле находится информация о жителях (тип структуры Liver) некоторого поселения: фамилия (непустая строка только из букв латинского алфавита), имя(непустая строка только из букв латинского алфавита), отчество(строка только из букв латинского алфавита; допускается пустая строка), дата рождения (в формате число, месяц, год), пол (символ ‘M’ - мужской символ ‘W’ - женский), средний доход за месяц (неотрицательное вещественное число). Напишите программу, которая считывает эту информацию из файла в односвязный упорядоченный список (в порядке увеличения возраста). Информация о каждом жителе должна храниться в объекте структуры Liver. Реализуйте возможности поиска жителя с заданными параметрами, изменение существующего жителя списка, удаления/добавления информации о жителях и возможность выгрузки данных из списка в файл (путь к файлу запрашивайте у пользователя с консоли). Добавьте возможность отменить $N/2$ последние введённых модификаций, то есть аналог команды Undo; N - общее количество. модификаций на текущий момент времени с момента чтения файла/последней отмены введённых модификаций.

Решение.

Общая идея решения:

Программа представляет собой систему управления информацией о жителях поселения, используя односвязный упорядоченный список. Данные о жителях считываются из файла, включая фамилию, имя, отчество, дату рождения, пол и средний доход. Каждый житель представлен объектом структуры Liver и упорядоченно включается в список по возрасту. Программа обеспечивает эффективное управление данными и обеспечивает удобный ввод информации о жителях из внешних источников.

Основные функции:

`print_liver:`

- Функция для вывода информации о жителе в файл.

`create_node:`

- Функция для создания узла списка по заданным данным о пациенте.

`add_node:`

- Функция для добавления узла в отсортированный список жителей по дню рождения.

`find_same_liver:`

- Функция для поиска узла в списке с теми же данными о жителе.

`free_list:`

- Функция для освобождения памяти, занятой списком жителей.

`undostack_push_back:`

- Функция для добавления узла в стек отката.

`delete_stack:`

- Функция для освобождения памяти, занятой стеком отката.

`copy_liver:`

- Функция для создания копии структуры жителей.

`print_undo_stack:`

- Функция для вывода информации о стеке отката в консоль.

`delete_node_list:`

- Функция для удаления узла из списка.

`interactive:`

- Основной цикл программы, который позволяет пользователям выбирать различные действия, включая добавление нового жителя, изменение данных о существующем жителе, удаление жителя, вывод информации в файл, поиск жителя, отмена последней операции, отображение информации о всех пациентах и выход из программы.

Использованные структуры и перечисления:

`Liver:`

- Структура, представляющая информацию о человеке, включая фамилию, имя, отчество, день рождения, пол и средний доход.

`node:`

- Структура, представляющая узел односвязного списка, содержащая указатель на структуру Liver и указатель на следующий узел.

`list:`

- Структура, представляющая односвязный список жителей, содержащая указатель на первый узел.

`type:`

- Перечисление для обозначения типа операции (удаление, добавление, модификация) в стеке отката (undo_stack).

`stack_node:`

- Структура, представляющая узел стека отката, содержащая указатель на узел списка, тип операции, и указатель на текущий узел в списке.

`undo_stack:`

- Структура, представляющая стек отката операций, содержащая массив узлов stack_node, размер стека и размер буфера.

`statements:`

- Перечисление описывающее типы ошибок, для их последующей обработки.

Данная программа на языке C представляет систему управления информацией о жителях поселения, используя структуры данных и файловый ввод/вывод. В ее функционал включены возможности поиска, изменения, удаления и добавления данных о жителях. Также предусмотрена поддержка отмены последних модификаций. Программа осуществляет валидацию вводимых данных и предоставляет пользователю удобный интерфейс в виде консольного меню.

Задание 8. [28]

На основе односвязного списка реализуйте тип многочлена от одной переменной (коэффициенты многочлена являются целыми числами). Реализуйте функции, репрезентирующие операции сложения многочленов, вычитания многочлена из многочлена, умножения многочленов, целочисленного деления многочлена на многочлен, поиска остатка от деления многочлен на многочлен, вычисления многочлена в заданной точке, нахождения производной многочлена, композиции многочленов. Для демонстрации работы вашей программы реализуйте возможность обработки текстового файла (с чувствительностью к регистру) следующего вида: Add(2x^2-x+2,-x^2+3x-1); % сложить заданные многочлены; Div(x^5,x^2-1); % разделить нацело заданные многочлены.

Возможные инструкции в файле: однострочный (начинается с символа ‘%’) комментарий, многострочный (начинается с символа ‘[‘, заканчивается символом ‘]’, вложенность запрещена) комментарий; Add – сложение многочленов, Sub - вычитание многочлена из многочлена, Mult – умножение многочленов, Div – целочисленное деление многочлена на многочлен, Mod – остаток от деления многочлена на многочлен, Eval – вычисление многочлена в заданной точке, Diff – дифференцирование многочлена, Cmps – композиция двух многочленов. Если в инструкции файла один параметр, то это означает, что вместо первого параметра используется текущее значение сумматора.

Например:

Mult(x^2+3x-1,2x+x^3); % умножить два многочлена друг на друга результат сохранить в сумматор и вывести на экран;

Add(4x-8); % в качестве первого аргумента будет взято значение из сумматора, результат будет занесен в сумматор;

Eval(1); % необходимо будет взять многочлен из сумматора и для него вычислить значение в 1.

Значение сумматора в момент начала выполнения программы равно 0

Решение

Общая идея решения:

Данный код реализует операции над многочленами от одной переменной с целочисленными коэффициентами, используя односвязные списки. Программа считывает команды из текстового файла и выполняет соответствующие операции над многочленами. Для каждой операции создана соответствующая функция.

Основные функции:

`statements:`
- Перечисление описывающее типы ошибок, для их последующей обработки.

`screate_polynom:`
- Создание нового полинома.

`clear_elements:`
- Очистка памяти, занимаемой элементами полинома.

`delete_polynom:`
- Удаление полинома и освобождение памяти.

`calculate_polynom:`
- Вычисление значения полинома для заданного значения x.

`diff_polynom:`
- Вычисление производной полинома.

`add_koefficient`
- Добавление коэффициента к полиному.

`resize_polynom:`
- Изменение размера полинома.

`subscription_polynoms:`
- Вычитание одного полинома из другого.

`summation_polynoms:`
- Сложение двух полиномов.

`multiply_polynoms:`
- Умножение двух полиномов.

`devide_polynoms:`
- Деление полинома на другой с получением частного и остатка.

`validate_polynom:`
- Проверяет строку s на корректность представления полинома.

`run:`
- Запускает основной процесс программы, читая команды из файла и выполняя операции над полиномами.

Структуры и перечисления:

`Element:`

- Структура, представляющая элемент (коэффициент) полинома.

`Polynom:`

- Структура, представляющая полином, состоящий из элементов. Основные функции.

`statements:`

- Перечисление описывающее типы ошибок, для их последующей обработки.

Данная программа на языке C читает команды из файла, разделяет их на токены и выполняет операции над многочленами в соответствии с полученными командами. Результаты операций выводятся на экран.

Задание 9. [29]

A) Реализуйте приложение для сбора статистических данных по заданному тексту. Результатом работы программы является информация о том, сколько раз каждое слово из файла встречается в данном файле (путь к файлу - первый аргумент командной строки). Слова в файле разделяются сепараторами (каждый сепаратор - символ), которые подаются как второй и последующие аргументы командной строки. В интерактивном диалоге с пользователем реализуйте выполнение дополнительных опций: вывод информации о том сколько раз заданное слово встречалось в файле (ввод слова реализуйте с консоли); вывод первых n наиболее часто встречающихся слов в файле (значение n вводится с консоли); поиск и вывод в контексте вызывающего кода в консоль самого длинного и самого короткого слова (если таковых несколько, необходимо вывести в консоль любое из них). Размещение информации о считанных словах реализуйте посредством двоичного дерева поиска. B) Для построенного дерева в пункте A реализуйте и продемонстрируйте работу функции поиска глубины данного дерева. C) Для построенного дерева в пункте A реализуйте функции сохранения построенного дерева в файл и восстановления бинарного дерева из файла. При этом восстановленное дерево должно иметь точно такую же структуру и вид, как и до сохранения. Продемонстрируйте работу реализованных функций

Решение.

Общая идея решения:

Приложение собирает статистические данные о частоте встречаемости слов в тексте, используя двоичное дерево поиска.

Пользователь может выполнять дополнительные опции, такие как поиск по слову, вывод наиболее часто встречающихся слов, поиск самого длинного и самого короткого слова, вывод глубины дерева, сохранение и загрузка дерева из файла.

Основные функции:

- `create_node`: Создает новый узел с заданным значением и возвращает указатель на него. Если создание строки или узла не удалось, возвращается NULL.

- `delete_node`: Рекурсивно удаляет все узлы в дереве, освобождая выделенную память.
- `add_node`: Добавляет узел в дерево. Если узел с таким значением уже существует, увеличивает счетчик. Возвращает correct при успешном выполнении.
- `find_node`: Ищет узел с заданным значением в дереве. Возвращает указатель на найденный узел через параметр resulting и соответствующий статус выполнения.
- `deep_node`: Возвращает глубину дерева, т.е., максимальное число уровней от корня до самого дальнего узла.
- `delete_nodes`: Удаляет массив узлов и освобождает память.
- `print_node`: Выводит информацию о узле в консоль.
- `print_nodes`: Рекурсивно выводит информацию о всех узлах дерева в консоль с учетом глубины.
- `trace_nodes`: Рекурсивно записывает информацию о всех узлах дерева в файл.
- `get_line`: Чтение строки из файла до встречи с заданными разделителями.
- `load_nodes`: Загрузка дерева из файла, ожидается, что файл имеет формат, где каждая строка представляет собой слово, за которым следует число раз которое это слово встречается в тексте.
- `validate`: Проверка входных аргументов программы.
- `validate_file`: Проверка формата файла перед его обработкой.
- `build_tree`: Построение дерева на основе входных параметров командной строки и файла.
- `get_int`: Получение целочисленного ввода от пользователя.
- `run`: Основная функция выполнения программы, обработка команд пользователя и взаимодействие с деревом.

Структуры и перечисления:

- `String` : Структура, представляющая строку.
- `Node` : Структура, представляющая узел дерева:
- `statements` : перечисление описывающее типы ошибок, для их последующей обработки.

Разработано приложение на языке программирования C для анализа текста с использованием двоичного дерева поиска. Пользователь имеет

возможность выполнять различные действия, такие как поиск по слову, вывод наиболее часто встречающихся слов, определение глубины дерева, а также сохранение и загрузка структуры из файла.

Задание 10. [30]

Реализуйте приложение для построения дерева скобочного выражения. На вход программе через аргументы командной строки подается путь к файлу, в котором содержится произвольное число строк. Каждая строка является корректным скобочным выражением. Ваша программа должна обработать каждое скобочное выражение из файла и вывести в текстовый файл (путь к файлу - аргумент командной строки) деревья скобочных записей в наглядной форме (форму вывода определите самостоятельно). Возникающие при работе программы деревья не обязаны быть бинарными.

Формат вывода не фиксирован и определяется удобством восприятия.

Решение

Общая идея решения:

Приложение строит деревья для скобочных выражений из входного файла и сохраняет их в текстовый файл в наглядной форме.

Основные функции:

- `create_node`: Создает новый узел с заданным значением и возвращает указатель на него. Узел может иметь несколько дочерних узлов. Функция также выделяет память для буфера дочерних узлов.
- `add_child`: Добавляет дочерний узел к родительскому. Если необходимо увеличить размер буфера, производится соответствующее расширение.
- `delete_node`: Освобождает память, выделенную для узла, его данных и буфера дочерних узлов.
- `delete_nodes`: Рекурсивно удаляет все узлы дерева, включая дочерние узлы.
- `print_nodes`: Рекурсивно выводит информацию о узлах дерева в консоль с учетом глубины.
- `fprint_nodes`: Рекурсивно записывает информацию о узлах дерева в файл с учетом глубины.
- `process_line`: Обрабатывает строку (bline) и создает из неё дерево узлов. Функция парсит символы строки и строит дерево согласно правилам вложенности. Результат возвращается через параметр result.

- `process_file`: Обрабатывает файл с именем path, создает деревья для каждой строки и записывает их в выходной файл с именем out_path. Функция также освобождает выделенную память после завершения работы.
- `validate_input`: Проверяет правильность ввода командной строки. В данном случае, ожидается наличие двух аргументов: имя входного файла и имя выходного файла.
- `run`: Основная функция программы, выполняющая последовательность шагов: проверка входных данных, обработка файла и вывод результата.

Структуры и перечисления:

- `String`: Структура, представляющая строку.
- `Node`: Структура, представляющая узел дерева:
- `statements`: перечисление описывающее типы ошибок, для их последующей обработки.

Разработан код для построения дерева скобочного выражения на основе строк из входного файла. Полученные деревья отображаются в наглядной форме в выходном файле. Программа также включает обработку ошибок, включая проблемы с выделением памяти и ошибки при открытии файлов.

Лабораторная работа № 4.

Задание 1. [31]

Реализуйте приложение для организации макрозамен в тексте. На вход приложению через аргументы командной строки подаётся путь к текстовому файлу, содержащему в начале набор директив #define, а далее обычный текст. Синтаксис директивы соответствует стандарту языка C:

#define <def_name> <value>

Аргументов у директивы нет, директива не может быть встроена в другую директиву. Ваше приложение должно обработать содержимое текстового файла, выполнив замены последовательностей символов <def_name> на <value>. Количество директив произвольно, некорректных директив нет, объём текста во входном файле произволен. В имени <def_name> допускается использование символов латинского алфавита (прописные и строчные буквы не отождествляются) и символов арабских цифр; значение <value> произвольно и завершается символом переноса строки или символом конца файла. Для хранения имен макросов и макроподстановок используйте хеш-таблицу размера HASHSIZE (начальное значение равно 128). Для вычисления хеш-функции интерпретируйте <def_name> как число, записанное в системе счисления с основанием 62 (алфавит этой системы счисления состоит из символов {0, …,9, A, …, Z, a, …, z}). Хеш-значение для <def_name> в рамках хеш-таблицы вычисляйте как остаток от деления эквивалентного для <def_name> числа в системе счисления с основанием 10 на значение HASHSIZE. Для разрешения коллизий используйте метод цепочек. В ситуациях, когда после модификации таблицы длины самой короткой и самой длинной цепочек в хеш-таблице различаются в 2 раза и более, пересобирайте хеш-таблицу с использованием другого значения HASHSIZE (логику модификации значения HASHSIZE продумайте самостоятельно) до достижения примерно равномерного распределения объектов структур по таблице. Оптимизируйте расчёт хэш-значений при пересборке таблицы при помощи кэширования.

Решение.

Общая идея решения:

Данное приложение предназначено для организации макрозамен в тексте на основе директив #define с использованием хэш-таблиц.

Основные функции:

- `get_hash`: Получает хеш-код для указанного ключа.
- `delete_table`: Удаляет хэш-таблицу и освобождает выделенную память.
- `create_hash_table`: Создает новую хеш-таблицу заданного размера.
- `statements insert`: Вставляет новый элемент в хэш-таблицу.
- `paste_to_string`: Заменяет все вхождения подстроки в строке на новую подстроку.
- `resize_map`: Меняет размер хэш-таблицы.
- `better_inserting`: Вставляет новый элемент в хэш-таблицу, а затем проверяет и, при необходимости, изменяет размер таблицы.
- `parsing_file`: Обрабатывает файл, вставляя значения из хеш-таблицы вместо соответствующих макросов #define.

Программа считывает текстовый файл, содержащий директивы #define, и обрабатывает его. В процессе выполнения макрозамен в остальном тексте используется хеш-таблица для хранения соответствий имен и значений макросов.

Задание 2. [32]

Реализуйте приложение – интерпретатор операций над целочисленными массивами. Приложение оперирует целочисленными массивами произвольной длины с именами из множества {A, B, …, Z}. Система команд данного интерпретатора (прописные и строчные буквы отождествляются):

● Load A, in.txt; - загрузить в массив A целые числа из файла in.txt (во входном файле могут произвольно присутствовать сепарирующие символы - пробелы, табуляции и переносы строк; также могут быть невалидные строковые репрезентации элементов массива);

● Save A, out.txt; - выгрузить элементы массива A в файл out.txt;

● Rand A, count, lb, rb; - заполнить массив A псевдослучайными элементами из отрезка в $[lb; lr]$ количестве count штук.

● Concat A, b; - сконкатенировать два массива A и B результат сохранить в массив A;

● Free(a); - очистить массив A и сопоставить переменную A с массивом из 0 элементов;

● Remove a, 2, 7; - удалить из массива a 7 элементов, начиная с элемента с индексом 2;

● Copy A, 4, 10, b; - скопировать из массива A элементы с 4 по 10 (оба конца включительно) и сохранить их в b;

● Sort A+; - отсортировать элементы массива A по неубыванию;

● Sort A-; - отсортировать элементы массива A по невозрастанию;

● Shuffle A; - переставить элементы массива в псевдослучайном порядке;

● Stats a; - вывести в стандартный поток вывода статистическую информацию о массиве A: размер массива, максимальный и минимальный элемент (и их индексы), наиболее часто встречающийся элемент (если таковых несколько, вывести максимальный из них по значению), среднее значение элементов, максимальное из значений отклонений элементов от среднего значения;

● Print a, 3; - вывести в стандартный поток вывода элемент массива A стоящий на позиции с номером 3;

● Print a, 4, 16; - вывести в стандартный поток вывода элементы массива A, начиная с 4 по 16 включительно оба конца;

● Print a, all; - вывести в стандартный поток вывод все элементы массива A.

Индексирование в массивах начинается с 0. Для сортировки массивов и реализации инструкции Shuffle используйте стандартную функцию qsort, свои реализации алгоритмов сортировки не допускаются. Предоставьте

текстовый файл с инструкциями для реализованного интерпретатора и продемонстрируйте его работу. Обработайте ошибки времени выполнения инструкций.

Решение

Общая идея решения:

Программа реализует простой интерпретатор операций над целочисленными массивами.

Основные функции:

- `create_array_s`: Выделение памяти под структуру Array и её данные, инициализация свойств.
- `free_array`: Освобождение выделенной памяти для массива данных в структуре Array.
- `_better_attoi`: Улучшенная версия atoi, преобразует строку в целое число и обрабатывает ошибки.
- `load_array_file`: Загрузка целых чисел из файла в структуру Array.
- print_element: Вывод значения элемента на указанной позиции в массиве Array.
- `print_range`: Вывод элементов в указанном диапазоне в массиве Array.
- `print_elements`: Вывод всех элементов массива Array.
- `dump_array_tofile`: Запись элементов массива в файл.
- `random_fill_array`: Заполнение массива случайными значениями в указанном диапазоне.
- `arr_cat`: Конкатенация элементов массива b к массиву a.
- `remove_elements`: Удаление n элементов, начиная с указанного индекса в массиве.
- `copy_elements`: Копирование элементов из массива a в массив b в заданном диапазоне.
- `compare_asc`: Функция сравнения для сортировки по возрастанию, используемая в qsort.
- `compare_des`: Функция сравнения для сортировки по убыванию, используемая в qsort.
- `sort_array`: Сортировка элементов массива в порядке возрастания или убывания.
- `random_compare`: Функция сравнения для случайного перемешивания, используемая в qsort.

- `shuffle_array_s`: Случайное перемешивание элементов массива.
- `find_max`: Поиск максимального значения и его индекса в массиве.
- `find_min`: Поиск минимального значения и его индекса в массиве.
- `find_most_common`: Поиск наиболее частого значения в массиве и количества его вхождений.
- `find_average`: Вычисление среднего значения элементов массива.
- `find_max_distance_average`: Нахождение максимального отклонения от среднего значения массива.
- `clear_all_arrays`: Освобождение памяти для всех массивов в переданном массиве указателей.
- `clean_up`: Освобождение ресурсов после выполнения операций и завершения работы программы.
- `print_stats`: Вывод статистики о массиве, такой как максимальное, минимальное значения, среднее и др.
- `parsing`: Парсинг команд из файла и выполнение соответствующих операций с массивами.

Программа включает в себя функции для работы с массивами, выполнения сортировок, поиска элементов и других операций. Каждая функция специализирована для выполнения конкретной задачи в контексте общей логики интерпретатора.

Задание 5. [33]

На вход приложению через аргументы командной строки подаются пути к текстовым файлам, содержащим арифметические выражения (в каждой строке находится одно выражение). Выражения в файлах могут быть произвольной структуры: содержать произвольное количество арифметических операций (сложение, вычитание, умножение, целочисленное деление, взятие остатка от деления, возведение в целую неотрицательную степень), круглых скобок (задают приоритет вычисления подвыражений). В вашем приложении необходимо для каждого выражения из каждого входного файла:

● проверить баланс скобок;

● построить обратную польскую запись выражения;

● вычислить значение выражения с использованием алгоритма вычисления выражения, записанного в обратной польской записи.

В результате работы приложения для каждого файла необходимо вывести в стандартный поток вывода путь к файлу, а также для каждого выражения из этого файла необходимо вывести:

● исходное выражение;

● обратную польскую запись для исходного выражения;

● значение выражения.

В случае обнаружения ошибки в расстановке скобок либо невозможности вычислить значение выражения, для каждого файла, где обнаружены вышеописанные ситуации, необходимо создать текстовый файл, в который выписать для каждого ошибочного выражения из исходного файла: само выражение, его порядковый номер в файле (индексация с 0) и причину невозможности вычисления. Для решения задачи используйте собственную реализацию структуры данных вида стек на базе структуры данных вида односвязный список. Предоставьте текстовый файл с выражениями для реализованного приложения и продемонстрируйте его работу. Обработайте ошибки времени выполнения инструкций.

Решение.

Общая идея решения:

Программа, написанная на языке C, является консольным приложением. При запуске она принимает в аргументах командной строки пути к текстовым файлам. Каждый из этих файлов содержит строки с

арифметическими выражениями. Для каждого выражения программа выполняет следующие шаги: проверяет баланс скобок, строит обратную польскую запись и вычисляет значение выражения с использованием алгоритма для обратной польской записи. Если в процессе выполнения обнаруживается ошибка (например, неверная расстановка скобок или невозможность вычисления), программа создает текстовый файл с информацией об ошибке для каждого файла, где были обнаружены ошибки.

Основные функции:

- `create_stack`: Создает новый стек и возвращает указатель на него.
- `push_back`: Добавляет узел в конец стека.
- `stack_pop`: Удаляет и возвращает верхний узел из стека.
- `create_node`: Создает новый узел и возвращает указатель на него.
- `delete_node`: Освобождает память, выделенную для узла.
- `bdelete_node`: Освобождает память, выделенную для узла, и, если узел требует очистки символа, освобождает память для символа.
- `bdelete_stack`: Удаляет стек, освобождая память для всех узлов и, если необходимо, для символов.
- `delete_stack`: Удаляет стек, освобождая память для всех узлов.
- `print_nodes`: Рекурсивно выводит содержимое узлов стека.
- `print_stack`: Выводит содержимое стека.
- `ending_symb`: Проверяет, является ли символ оператором или скобкой.
- `scan_line`: Считывает строку из файла, пропуская пробелы, и возвращает указатель на динамически выделенную строку.
- `is_number`: Проверяет, является ли строка числом.
- `my_split`: Разделяет строку на подстроки, используя разделитель, и возвращает массив указателей на динамически выделенные строки.
- `eval`: Выполняет арифметическую операцию между двумя числами.
- `cleanup_memory`: Освобождает память, выделенную под массив строк, стек и выходную строку.
- `solve`: Преобразует инфиксное выражение в обратную польскую запись и записывает результат в выходную строку.
- `bfree`: Освобождает память, выделенную под массив строк.
- `calculate`: Вычисляет значение выражения, представленного в обратной польской записи.

Программа, написанная на языке C, представляет собой консольное приложение. Её основной функционал заключается в обработке

арифметических выражений из текстовых файлов. В процессе выполнения, она проводит проверку баланса скобок, строит обратную польскую запись и вычисляет значения выражений. В случае обнаружения ошибок, для каждого выражения создаются текстовые файлы с подробной информацией об ошибке. Реализация включает использование структур данных в виде стеков, построенных на основе односвязных списков.

Задание 6. [34]

Реализуйте приложение, которое по заданной булевой формуле строит таблицу истинности, описывающую логическую функцию, заданную формулой. Через аргументы командной строки приложению подается путь к файлу, который содержит одну строку, в которой записана булева формула. В этой формуле могут присутствовать:
- односимвольные имена логических переменных;
- константы 0 (ложь) и 1 (истина);
- & - оператор логической конъюнкции;
- | - оператор логической дизъюнкции;
- ~ - оператор логической инверсии;
- -> - оператор логической импликации;
- +> - оператор логической коимпликации;
- <> - оператор логического сложения по модулю 2;
- = - оператор логической эквиваленции;
- ! - оператор логического штриха Шеффера;
- ? - оператор логической функции Вебба;
- операторы круглых скобок, задающие приоритет вычисления подвыражений. Вложенность скобок произвольна. Для вычисления булевой формулы постройте бинарное дерево выражения и вычисление значения булевой формулы на конкретном наборе переменных выполняйте с помощью этого дерева. Приоритеты операторов (от высшего к низшему): 3(~), 2(?,!,+>,&), 1(|, ->, <>, =). В результате работы приложения необходимо получить выходной файл (имя файла псевдослучайно составляется из букв латинского алфавита и символов арабских цифр, файл должен быть размещён в одном каталоге с исходным файлом) с таблицей истинности булевой функции, заданной входной булевой формулой.

Решение.

Общая идея решения:

Эта программа, написанная на языке C, представляет консольное приложение. Её основной функционал заключается в построении таблицы истинности для заданной булевой формулы. Исходная булева формула представлена в инфиксной записи и преобразуется в постфиксную запись. Затем по этой записи строится дерево выражения. Для каждой возможной комбинации значений переменных из формулы программа вычисляет

результат логического выражения и выводит соответствующую таблицу истинности.

Основные функции:

- `is_operator`: Проверяет, является ли символ оператором.
- `print_var`: Печатает переменные в файл.
- `print_infix`: Рекурсивно печатает выражение в инфиксной форме.
- `print_header`: Печатает заголовок таблицы истинности, включая переменные и выражение.
- `priority`: Возвращает приоритет оператора.
- `is_two_char_operator`: Проверяет, является ли оператор двухсимвольным.
- `not_in_array`: Проверяет, отсутствует ли символ в массиве.
- `cnt_unique_lets`: Считает количество уникальных переменных в выражении.
- `generate_filename`: Генерирует псевдослучайное имя файла.
- `solution_from_tree`: Рекурсивно вычисляет значение выражения по дереву.
- `build_table`: Строит таблицу истинности и записывает её в файл.
- `infix_to_postfix`: Преобразует инфиксное выражение в постфиксное.
- `build_tree`: Строит дерево выражения на основе постфиксной записи.
- `run`: Запускает обработку файла с выражением.
- `exit_state`: Выводит сообщение в зависимости от статуса выполнения программы.
- `Stack_node`: Представляет собой один узел стека.
- `Stack`: Представляет собой структуру стека с указателем на вершину.
- `create_stack`: Инициализирует стек.
- `is_empty_stack`: Проверяет, пуст ли стек.
- `push_back_stack`: Помещает элемент в стек.
- `pop_stack`: Извлекает элемент из стека.
- `clear_stack`: Очищает весь стек.

Программа представляет собой консольное приложение на языке C, предназначенное для обработки арифметических выражений из текстовых файлов. Она проверяет баланс скобок, строит обратную польскую запись и вычисляет значения выражений. В случае ошибок создаются текстовые

файлы с информацией об ошибке для каждого выражения. Используются структуры данных в виде стеков на основе односвязных списков.

Задание 7. [35]

Опишите тип структуры MemoryCell, содержащей имя переменной и её целочисленное значение.

Через аргументы командной строки в программу подается файл с инструкциями вида

```
myvar=15;
bg=25;
ccc=bg+11;
print ccc;
myvar=ccc;
bg=ccc*myvar;
print;
```

Файл не содержит ошибок и все инструкции корректны. В рамках приложения реализуйте чтение данных из файла и выполнение всех простых арифметических операций (сложение, вычитание, умножение, целочисленное деление, взятие остатка от
деления), инициализации переменной и присваивания значения переменной (=) и операции print (вывод в стандартный поток вывода либо значения переменной, имя которой является параметром операции print, либо значений всех объявленных на текущий момент выполнения переменных с указанием их имён). В каждой инструкции может присутствовать только одна из вышеописанных операций; аргументами инструкций могут выступать значение переменной, подаваемое в виде имени этой переменной, а также целочисленные константы, записанные в системе счисления с основанием 10. При инициализации переменной по необходимости требуется довыделить память в динамическом массиве структур типа MemoryCell; инициализация переменной (по отношению к операции перевыделения памяти) должна выполняться за амортизированную константу. Для поиска переменной в массиве используйте алгоритм дихотомического поиска, для этого ваш массив в
произвольный момент времени должен находиться в отсортированном состоянии по ключу имени переменной; для сортировки массива используйте стандартную функцию qsort, реализовывать непосредственно какие-либо алгоритмы сортировки запрещается. Имя переменной может иметь произвольную длину и содержать только символы латинских букв, прописные и строчные буквы при этом не отождествляются. В случае использования в вычислениях не объявленной переменной необходимо

остановить работу интерпретатора и вывести сообщение об ошибке в стандартный поток вывода. Предоставьте текстовый файл с инструкциями для реализованного интерпретатора и продемонстрируйте его работу. Обработайте ошибки времени выполнения инструкций.

Общая идея решения:

Данный код представляет из себя простой интерпретатор, способный выполнять базовые арифметические операции, операции присваивания и выводить значения переменных.

Основные функции:

- `ending_symb`: Проверяет, является ли символ одним из символов завершения операции.
- `scan_lexema`: Считывает лексему из файла до символа завершения, динамически выделяя память и автоматически увеличивая размер буфера при необходимости. Возвращает указатель на считанную лексему и символ завершения.
- `run`: Открывает файл по указанному пути, использует scan_lexema для считывания лексем из файла, выполняет операции с ячейками памяти в соответствии с содержимым файла, вызывает функции для работы с ячейками памяти.
- `create_stack`: Инициализирует стек.
- `is_empty_stack`: Проверяет, пуст ли стек.
- `push_back_stack`: Добавляет элемент в стек.
- `pop_stack`: Извлекает элемент из стека.
- `clear_stack`: Очищает стек.
- `create_node`: Создает узел с данными.
- `free_node`: Очищает узел.
- `free_tree`: Очищает дерево.
- `node_stack_init`: Инициализирует стек узлов.
- `is_node_stack_empty`: Проверяет, пуст ли стек узлов.
- `push_node`: Добавляет узел в стек узлов.
- `pop_node`: Извлекает узел из стека узлов.
- `clear_node_stack`: Очищает стек узлов.
- `is_latin`: Проверяет, состоит ли строка только из латинских букв.
- `is_number`: Проверяет, является ли строка числом.
- `bin_search`: Бинарный поиск в массиве ячеек памяти по ключу.
- `create_cell`: Создает ячейку памяти с ключом и значением.

- `add_cell`: Добавляет ячейку памяти в массив ячеек.

Данный код представляет собой простой интерпретатор, который может выполнять базовые арифметические операции, операции присваивания и выводить значения переменных.

Заключение

Данный курсовой проект был полезным опытом разработки на языке Си, в ходе которого я изучил и на практике применил множество важных концепций и технологий в области программирования. Вот самые основные моменты, которые я смог вынести для себя после выполнения этой работы:

1. Взаимодействие с файлами:
    - Возможность взаимодействия с файлами и чтение инструкций из текстового файла обогатило мой опыт обработки файлов в среде C, что, безусловно, полезно при реализации практических решений.
2. Обработка ошибок:
    - Поскольку каждое решение задачи требовало внимательного подхода к обработке ошибок, мой навык обработки ошибок значительно усилился. Создание более надежных программ стало не только целью, но и результатом данной работы.
3. Теоретические знания в программировании:
    - Так как по ходу работы, необходимо было применять разные структуры данных и алгоритмы на практике, то пришлось также и изучить теорию.
4. Практический опыт в программировании:
    - Проект обеспечил мне обширный практический опыт в различных аспектах программирования, начиная от проектирования структур данных и заканчивая реализацией алгоритмов. Этот опыт является ценным активом для моего профессионального роста, а также неплохим портфолио для дальнейшей деятельности.
5. Улучшение навыков отладки и тестирования:
    - Процесс разработки приложений поднял мой профессиональный уровень в сфере отладки и тестирования. В моём арсенале появился такой инструмент как valgrind.

Общий вывод:

Выполнение курсовой работы принесло мне значительный опыт в различных аспектах программирования, начиная от языка C и заканчивая обработкой файлов и управлением ошибками. Мои навыки в разработке программ значительно усовершенствовались, и я более глубоко разобрался в принципах работы структур данных и алгоритмов. Помимо этого, я расширил свой набор методов отладки и тестирования программного кода.

Этот проект стал важным этапом в моем обучении, и я уверен, что полученные знания и опыт помогут в моих будущих проектах и при решении различных задач, даже в других областях, ведь был получен не только лишь опыт разработки на Си, а также навык решения проблем, которые возникали по ходу работы. Надеюсь, что в будущем меня будут ждать ещё более интересные задачи, которые я также буду успешно решать.

Список использованных источников

1. Практические занятия и лекции по Математическому практикуму

Приложение

[1]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_1/work_1

[2]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_1/work_2

[3]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_1/work_3

[4]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_1/work_4

[5]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_1/work_5

[6]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_1/work_6

[7]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_1/work_7

[9]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_1/work_9

[10]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_1/work_10

[11]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_2/work_1

[12]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_2/work_2

[13]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_2/work_3

[14]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_2/work_4

[15]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_2/work_5

[16]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_2/work_6

[17]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_2/work_7

[18]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_2/work_8

[19]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_2/work_9

[20]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_2/work_10

[21]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_3/1

[22]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_3/2

[23]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_3/3

[24]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_3/4

[25]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_2/5

[26]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_3/6

[27]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_3/7

[28]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_3/8

[29]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_3/9

[30]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_3/10

[31]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_4/1

[32]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_4/2

[33]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_4/5

[34]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_4/6

[35]https://github.com/yashelter/mathematical_practical/tree/main/solutions/lab_4/7