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»

Тема: «Разработка прикладных программ на языке Си для решения практических задач»

Студент: Заславцев М. В.

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.

Работа № 1.

Задание № 1.1 [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 суть задачи:

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:

`is_natural`:

- Проверяет, является ли переданное значение натуральным числом.

`overflow_underflow_normal`:

- Проверяет на переполнение, недополнение или нормальное значение числа.

`is_digit`:

- Проверяет, является ли переданная строка десятичным числом.

`factorial`:

- Рекурсивно вычисляет факториал числа.

`sum_of_natural_numbers`:

- Вычисляет сумму натуральных чисел от 1 до переданного числа.

`is_negative`:

- Проверяет, является ли переданная строка отрицательным числом.

`is_negative_null`:

- Проверяет, является ли число отрицательным и равным 0, при условии изменения строки на "0" в случае соответствия.

`divide_to_digits`:

- Разделяет число на отдельные цифры и сохраняет их в массив.

`is_prime`:

- Проверяет, является ли число простым.

`find_multiples`:

- Ищет все кратные заданному делителю в пределах заданного диапазона.

`table_of_degrees`:

- Выводит таблицу степеней для чисел от 1 до 10 с заданным диапазоном степеней.

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 № 1.2 [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	$\gamma = \lim\limits_{m\to\infty}\left(\sum\limits_{k=1}^{m} C_m^k \frac{(-1)^k}{k}\ln(k!)\right)$	$\gamma = -\frac{\pi^2}{6} + \sum\limits_{k=2}^{\infty}\left(\frac{1}{\lfloor\sqrt{k}\rfloor^2} - \frac{1}{k}\right)$	$e^{-x} = \lim\limits_{t\to\infty}\left(\ln t \prod\limits_{p\le t, p\in P}\frac{p-1}{p}\right)$

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

Решение

Общая суть задачи:

Общая суть задачи заключается в создании программы на языке C, которая вычисляет значения математических констант e, π, $ln\ 2$, sqrt(2), и γ с заданной точностью. Для каждой константы предоставлены три способа вычисления: через сумму ряда, решение специального уравнения и значение предела.

Основные функции:

Вычисление e (число Эйлера):

- `Euler_number_by_limit`: Вычисление через предел.
- `series_Euler_Number`: Вычисление через сумму ряда.
- `function_lnX_1, dfunction_ln`: Решение уравнения $ln(x) = 1$ методом Ньютона.

Вычисление π (число Пи):

- `Pi_number_by_limit`: Вычисление через предел.

- `series_pi_Number`: Вычисление через сумму ряда.
- `function_cosx_1, dfunction_cos`: Решение уравнения $\cos(x) + 1 = 0$ методом Ньютона.

Вычисление $ln\ 2$ (натуральный логарифм от 2):

- `ln_number_by_limit:` Вычисление через предел.
- `series_ln_Number`: Вычисление через сумму ряда.
- `function_ex_2, dfunction_ex`: Решение уравнения $e\text{^}x = 2$ методом Ньютона.

Вычисление √2 (квадратный корень из 2):

- `sqrt_number_by_limit:` Вычисление через предел.
- `Product_sqrt_Number:` Вычисление через произведение.
- `function_x2_2, dfunction_x2`: Решение уравнения $x\text{^}2 = 2$ методом Ньютона.

Вычисление γ (постоянная Эйлера-Маскерони):

- `gamma_number_by_limit:` Вычисление через предел.
- `series_gamma_Number:` Вычисление через сумму ряда.
- `function_e_gamma, dfunction_e_gamma:` Решение уравнения $e\text{^}{-}x = \lim(ln(t) \prod (p - 1) / p)$ методом Ньютона.

Данная программа решает задачу вычисления значений математических констант (e, π, $ln\ 2$, √2, γ) с заданной точностью, используя различные методы, такие как сумма ряда, решение специального уравнения и значение предела. Каждая константа имеет три реализации вычислительных методов. Программа предоставляет гибкую возможность выбора точности вычислений, предоставляя пользователю управление этим параметром через аргумент командной строки.

Задание 1.3 [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 суть задачи:

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:

`Input_checker`:
- Проверяет корректность ввода, определяет флаг и параметры в зависимости от количества аргументов и их типов.

`use_flag`:
- Вызывает соответствующую функцию в зависимости от переданного флага, передавая ей необходимые параметры.

`solution_equation`:
- Решает квадратное уравнение с заданными коэффициентами.

`multiplicity_of_two_numbers`:
- Проверяет, является ли одно число кратным другому.

`is_right_triangle`:
- Проверяет, могут ли три числа являться длинами сторон прямоугольного треугольника.

`print_triangle_solution`:
- Выводит результат проверки на возможность образования прямоугольного треугольника.

`print_solution_the_equation`:
- Выводит решение квадратного уравнения.

`check_epsilon`:
- Проверяет корректность значения эпсилон.

`flag_checker`:
- Проверяет корректность ввода флага.

is_digit:

- Проверяет, является ли строка числом.

`overflow_underflow_normal`:

- Проверяет на переполнение, недополнение или нормальное значение числа.

Эти функции обеспечивают выполнение задач, описанных в постановке, такие как решение квадратного уравнения, проверка кратности чисел, и определение возможности образования прямоугольного треугольника.

Задание 1.4 [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 суть задачи:

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:

`create_output_file`:
- Создает имя выходного файла на основе имени входного файла.

`check_flags`:
- Проверяет наличие опционального символа 'n' во втором символе флага.

`input_checker`:
- Проверяет корректность ввода и определяет действие программы в зависимости от флага.

`file_read_write`:
- Проверяет, можно ли открыть файл для чтения или записи.

`without_arabic`:
- Исключает символы арабских цифр из входного файла.

`count_of_latin_or_not_latin`:
- Подсчитывает количество латинских символов или отличных от латинских символов, арабских цифр и пробела в каждой строке входного файла.

`transfer_to16`:
- Переводит символ в его ASCII-код, представленный в системе счисления с основанием 16.

`liters_to_16ss`:
- Заменяет символы, отличные от цифр, их ASCII-кодами в системе счисления с основанием 16.

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 1.5 [5]

Вычислить значения сумм с точностью ε, где 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. $\sum_{n=1}^{\infty} \frac{(-1)^n (2n-1)!! x^{2n}}{(2n)!!}$

Решение

Общая суть задачи:

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:

`check_epsilon`:

- Проверяет корректность введенного значения эпсилон.

`overflow_underflow_normal`:

- Проверяет переполнение, недополнение или нормальное значение для целочисленных переменных.

`is_digit`:

- Проверяет, является ли строка числом и возвращает его значение.

`flag_checker`:

- Проверяет корректность флага.

`Input_checker`:

- Проверяет корректность ввода, включая флаги, значения и эпсилон.

`print_sum_`:

- Выводит результат вычисления суммы.

`sum_a, sum_b, sum_c, sum_d:`

- Реализуют вычисление суммы для четырех различных функций с использованием заданной точности.

Программа проверяет корректность введенных данных, включая значения эпсилон и флаги. Затем она вычисляет значения сумм для каждой из четырех функций и выводит результаты.

Задание 1.6 [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. $\int_0^1 x^x dx$

Решение

Общая суть задачи:

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

Основные функции:

`check_flags`:
- Проверяет флаги командной строки для вывода информации или помощи.

`check_epsilon`:
- Проверяет корректность введенного значения эпсилон.

`is_digit`:
- Проверяет, является ли строка числом и возвращает его значение.

`Input_checker`:
- Проверяет корректность ввода, включая значения эпсилон и флаги.

`print_integral_sum`:
Выводит результат вычисления интеграла.

`func_1`, `func_2`, `func_3`, `func_4`:
- Реализуют четыре различные функции, для которых нужно вычислить интеграл.

`simpson_method`:
- Реализует метод Симпсона для вычисления интеграла с заданной точностью.

Программа проверяет корректность введенных данных, включая значения эпсилон. Затем она вычисляет значения интегралов для каждой из четырех функций с использованием метода Симпсона и выводит результаты.

Задание 1.7 [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 суть задачи:

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_trashes`:
- Перемещает указатель файла на первый непустой символ.

`is_latin_ch`:
- Проверяет, является ли символ буквой латинского алфавита.

`str_tolower`:
- Преобразует строки в нижний регистр.

`get_str`:
- Считывает строку из файла до пробельного символа или конца строки.

`read_write_files_r`:
- Считывает лексемы из двух файлов и записывает их в третий файл в соответствии с условиями задачи.

`flag_r`:
- Обработка флага -r, вызывает функцию `read_write_files_r`.

`transfer_to_crnt_base` / `to_ascii_and_4base`:
- Преобразует числа в систему счисления с основанием 4 или 8.

`read_write_files_a`:
- Считывает лексемы из файла и преобразует их в соответствии с условиями задачи.

`flag_a`:
- Обработка флага -a, вызывает функцию `read_write_files_a`.

`input_checker`:
- Проверяет корректность ввода аргументов командной строки и вызывает соответствующую обработку в зависимости от флага.

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 1.8 [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 суть 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_to_num`:

- Переводит символ в соответствующую цифру в системе счисления.

`ss_to_base_10`:

- Преобразует число из любой системы счисления в десятичную.

check_symbol:

- Проверяет, является ли символ допустимым для числа.

`find_base`:

- Определяет основание системы счисления для символа.

`Input_checker`:

- Проверяет корректность ввода аргументов командной строки и открывает входной файл.

Программа обрабатывает файл с числами, определяет для каждого числа минимальное основание системы счисления и выводит результаты в указанный выходной файл. Ошибки, такие как переполнение и некорректные числа, также обрабатываются.

Задание 1.9 [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

Общая суть задачи:

Поменять местами максимальный и минимальный элементы массива:

- Заполнить массив фиксированного размера псевдослучайными числами в заданном диапазоне.
- Найти максимальный и минимальный элементы в массиве.
- Поменять местами найденные элементы за один проход по массиву.
- Вывести исходный и измененный массивы.

Формирование массива C из массивов A и B:

- Заполнить динамические массивы A и B псевдослучайными числами в заданных диапазонах.
- Сформировать массив C, где каждый элемент i равен сумме i-го элемента массива A и ближайшего к нему элемента из массива B.
- Вывести массивы A, B и C на экран.

Программа принимает аргументы командной строки для определения диапазонов генерации случайных чисел. Задача 1 выполняется при наличии двух аргументов, а задача 2 - при их отсутствии.

Основные функции:

Задача 1: Поменять местами максимальный и минимальный элементы массива

`swap_maxmin_elements`:

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

`compare`:

- Функция для использования в qsort для сортировки массива B перед формированием массива C.

`bin_search`:

- Функция для бинарного поиска ближайшего значения к A[i] в массиве B.

Задача 2: Формирование массива C из массивов A и B

`generate_c`:

- Функция, формирующая 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.

`len_array1`, `len_array2`:

- Длины массивов A и B генерируются псевдослучайно в заданном диапазоне [10..10000].

`array_a`, `array_b`, `array_c`:

- Динамические массивы для хранения значений.

`fill_array_with_random_numbers`:

- Заполняет массивы A и B псевдослучайными числами в диапазоне [-1000..1000].

`qsort`:

- Сортирует массив B перед использованием в бинарном поиске.

`print_array`:

- Функция для вывода содержимого массива на экран.

Данная программа представляет собой универсальный инструмент, способный выполнять две задачи:

В рамках первой задачи программа генерирует массив фиксированного размера, заполняя его случайными числами в указанном пользователем диапазоне. Затем она находит минимальное и максимальное значения в массиве и меняет их местами за один проход по массиву.

Вторая задача включает генерацию двух динамических массивов различной длины и заполнение их случайными числами. Затем программа формирует третий массив, где каждый элемент равен сумме соответствующего элемента из первого массива и ближайшего к нему элемента из второго массива.

Таким образом, программа демонстрирует возможность обработки массивов различных типов и размеров, предоставляя универсальный функционал для выполнения задач по их модификации.

Задание 1.10 [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 суть задачи:

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:

`check_start_enter`:

- Проверяет корректность введенного основания системы счисления.

`char_to_num`:

- Переводит символ в соответствующую цифру в системе счисления.

`ss_to_base_10`:

- Преобразует строку числа из системы счисления с заданным основанием в десятичное число.

`transfer_from_int`:

- Преобразует десятичное число в строковое представление в системе счисления с заданным основанием.

`print_to_radix`:

- Выводит строковое представление числа в указанной системе счисления.

Программа также обрабатывает случай, когда вводится строка "Stop" для завершения ввода чисел. В случае ошибок, таких как неверное основание системы счисления или некорректные числа, программа выводит соответствующие сообщения.

Работа № 2.

Задание 2.1 [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 суть задачи:

Программа предназначена для обработки строк в зависимости от переданных аргументов командной строки.

Основные функции:

`_strcmp`:
Сравнивает две строки и возвращает `true`, если они равны, и `false` в противном случае.
`swap`:

- Обменивает две строки.

`is_alpha`:

- Проверяет, является ли символ буквой.

`is_digit`:

- Проверяет, является ли символ цифрой.

`toapper`:

- Преобразует символ в верхний регистр.

`is_number`:

- Проверяет, состоит ли строка только из цифр.

`string_to_unsigned_int`:

- Преобразует строку в беззнаковое целое число.

`string_length`:

- Возвращает длину строки.

`reverse`:

- Создает новую строку, являющуюся перевернутой версией входной строки.

`convert_flag_u`:

- Создает новую строку, в которой символы на нечетных позициях в верхнем регистре.

`convert_flag_n`:

- Создает новую строку, разделяя символы на цифры, буквы и остальные.

`convert_flag_c`:

- Создает новую строку, конкатенируя строки в псевдослучайном порядке.

Данная программа предназначена для обработки строк в зависимости от флага, переданного через аргументы командной строки. Реализованы функции для подсчета длины строки, создания перевернутой строки, преобразования строки с заменой символов на нечетных позициях в верхний регистр, формирования новой строки с сортировкой символов по типам (цифры, буквы, прочие) и конкатенации строк в псевдослучайном порядке.

Программа поддерживает ввод через командную строку с использованием флагов и аргументов, а также возвращает результаты в динамически выделенной памяти.

Задание 2.2 [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 суть задачи:

Программа включает две функции: одну для вычисления среднего геометрического введенных чисел и другую для рекурсивного возведения вещественного числа в целую степень с использованием алгоритма быстрого возведения в степень. Вот общая суть программы и краткое описание функций:

Основные функции:

Функция `fpow` (Fast Power):

- Реализует алгоритм быстрого возведения в степень для вещественных чисел.
- Рекурсивно вычисляет a^n, используя свойство a^(2n) = (a^n)^2 и a^(2n+1) = a * (a^n)^2.
- Если степень отрицательна, возвращает обратное значение результата, чтобы обеспечить корректное вычисление.

Функция `geometry_aver`:

- Принимает переменное число аргументов вещественного типа и их количество.

- Использует библиотечную функцию `va_start`, `va_arg`, и `va_end` для обработки переменного числа аргументов.
- Вычисляет произведение всех аргументов и затем извлекает корень n-ной степени (где n - количество аргументов) из этого произведения.
- Возвращает среднее геометрическое переданных чисел.

Программа включает две функции: `fpow` для быстрого возведения вещественного числа в целую степень и `geometry_aver` для вычисления среднего геометрического переменного числа вещественных аргументов. Функция `fpow` использует алгоритм быстрого возведения в степень, а `geometry_aver` принимает переменное число аргументов и возвращает среднее геометрическое.

Задание 2.3 [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 суть задачи:

Программа включает в себя функцию, реализующую поиск подстроки в содержимом файлов. Запрещено использовать стандартные функции strstr и strncmp из библиотеки string.h. Приведены также структуры данных и функции для обработки данной задачи. Вот общая суть программы и краткое описание функций:

Основные функции:

`create_shift_table`:

- Создает таблицу сдвигов для алгоритма Кнута-Морриса-Пратта.
- Используется для эффективного поиска подстроки в тексте.

Функция `find_patterns`:

- Принимает переменное количество аргументов, представляющих пути к файлам, и подстроку для поиска.
- Для каждого файла выполняет поиск всех вхождений подстроки в его содержимое.
- Результаты (индексы и номера строк) сохраняются в структуре `Indexes`.
- Проверяет наличие ошибок при открытии файлов, выделении памяти и других сценариях.
- Выводит наглядный результат: для каждого файла и каждого вхождения подстроки выводит номер строки и номер символа в строке, с которого начинается вхождение.

Программа включает функцию поиска подстроки в содержимом файлов, без использования стандартных функций `strstr` и `strncmp`. Она использует алгоритм Кнута-Морриса-Пратта для эффективного поиска. Результаты, такие как индексы и номера строк, хранятся в структуре `Indexes`. Функция `find_patterns` принимает пути к файлам и подстроку, выполняет поиск, и выводит информацию о вхождениях в каждом файле.

Задание 2.4 [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 суть задачи:

Программа включает в себя функцию, реализующую поиск подстроки в содержимом файлов. Запрещено использовать стандартные функции strstr и strncmp из библиотеки string.h. Приведены также структуры данных и функции для обработки данной задачи. Вот общая суть программы и краткое описание функций:

Основные функции:

`create_shift_table`:

- Создает таблицу сдвигов для алгоритма Кнута-Морриса-Пратта.
- Используется для эффективного поиска подстроки в тексте.

Функция `find_patterns`:

- Принимает переменное количество аргументов, представляющих пути к файлам, и подстроку для поиска.
- Для каждого файла выполняет поиск всех вхождений подстроки в его содержимое.
- Результаты (индексы и номера строк) сохраняются в структуре `Indexes`.
- Проверяет наличие ошибок при открытии файлов, выделении памяти и других сценариях.
- Выводит наглядный результат: для каждого файла и каждого вхождения подстроки выводит номер строки и номер символа в строке, с которого начинается вхождение.

Программа включает функцию поиска подстроки в содержимом файлов, без использования стандартных функций `strstr` и `strncmp`. Она использует алгоритм Кнута-Морриса-Пратта для эффективного поиска. Результаты, такие как индексы и номера строк, хранятся в структуре `Indexes`. Функция `find_patterns` принимает пути к файлам и подстроку, выполняет поиск, и выводит информацию о вхождениях в каждом файле.

Задание 2.5 [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 суть задачи:

Программа включает в себя реализацию функций для работы с различными числовыми системами, вычислением многочленов, определением выпуклости многоугольников, а также функции для вывода информации в различных форматах.

Основные функции:

`int_transfer_to_roman`:

- Переводит целое число в римскую систему счисления.

`transfer_cyckendorf`:

- Переводит целое число в представление чисел Циккендорфа.

`int_transfer_to_base`:

- Переводит целое число в указанную систему счисления и возвращает строку с результатом.

`ss_to_base_10`:

- Переводит строку, представляющую число в определенной системе счисления, в десятичное число.

`print_memory_dump`:

- Выводит в двоичной системе счисления представление числа в памяти.

`overfprintf`:

- Переопределение функции `fprintf` для обработки пользовательских флагов в формате строки.

`oversprintf`:

- Переопределение функции `sprintf` для обработки пользовательских флагов в формате строки.

Программа обеспечивает удобные средства для работы с числовыми представлениями, многочленами и геометрическими фигурами, а также предоставляет возможность форматированного вывода информации в различных форматах.

Задание 2.6 [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 суть задачи:

Общая суть предоставленного кода заключается в реализации пользовательских функций форматированного ввода overfscanf и форматированного вывода oversprintf, аналогичных стандартным функциям fscanf и sprintf, соответственно. В этих функциях добавлены дополнительные флаги для обработки специальных форматов данных.

Основные функции:

`roman_to_int`:

- Преобразует строку, представляющую римское число, в целое число типа `int`.

`transfer_cyckendorf_to_int`:

- Преобразует строку, представляющую число в цекендорфовой системе, в целое число типа `unsigned int`.

`ss_to_base_10`:

- Преобразует строку, представляющую число в системе счисления с заданным основанием, в целое число типа `int`.

`print_errors`:

- Выводит сообщение об ошибке в зависимости от переданного статуса.

`process_user_flags`:

- Обрабатывает пользовательские флаги в форматной строке и выполняет соответствующие операции с входными данными.

`overfscanf`:

- Функция, аналогичная стандартной `fscanf`, но с поддержкой пользовательских флагов для дополнительных форматов.

`oversscanf`:

- Функция, аналогичная стандартной `sscanf`, но с поддержкой пользовательских флагов для дополнительных форматов.

Обе функции `overfscanf` и `oversscanf` обрабатывают пользовательские флаги в форматной строке и осуществляют считывание данных из потока ввода или строки с учетом дополнительных форматов, таких как римские числа, цекендорфова система и система счисления с заданным основанием. Функции возвращают количество успешно обработанных пользовательских флагов.

Задание 2.7 [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 суть задачи:

Необходимо реализовать функцию для нахождения корня уравнения одной переменной методом дихотомии. Функция должна принимать границы интервала, точность (эпсилон) и указатель на функцию, описывающую уравнение.

Основные функции:

`func_1` и `func_2`:

- Это функции, представляющие уравнения, для которых мы хотим найти корни.

`dichotomy`:

- Функция, реализующая метод дихотомии для нахождения корня уравнения на заданном интервале с заданной точностью. Итеративно делят интервал пополам и выбирают тот подинтервал, на котором изменение знака функции гарантирует наличие корня. Возвращает найденное значение корня.

Функция `dichotomy` применяется для нахождения корней заданных уравнений, и результаты демонстрируются на примерах.

Задание 2.8 [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. Продемонстрируйте работу функции.

Решение

Общая суть задачи:

Необходимо реализовать функцию с переменным числом аргументов для вычисления суммы переданных целых неотрицательных чисел в заданной системе счисления с основанием от 2 до 36. Также требуется реализовать функцию для "сложения в столбик" двух чисел в заданной системе счисления.

Основные функции:

`remove_zeros`:

- Функция удаляет ведущие нули из строки числа.

`sum2nums`:

- Функция выполняет сложение двух чисел в заданной системе счисления методом "сложения в столбик". Возвращает строку, представляющую сумму.

`sum_in_crnt_base`:

- Функция с переменным числом аргументов для вычисления суммы переданных чисел в заданной системе счисления. Внутри использует `sum2nums` для поочередного сложения чисел. Удаляет ведущие нули в результирующей строке.

  Функции позволяют работать с числами в различных системах счисления и эффективно выполнять их сложение.

Задание 2.9 [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 суть задачи:

Необходимо реализовать функцию с переменным числом аргументов, которая для каждой переданной десятичной дроби в виде значения вещественного типа в диапазоне (0; 1) определяет, имеет ли она в системе счисления с заданным основанием конечное представление.

Основные функции:

`print_arr`:

- Функция для вывода информации о числах из массива структур `Numb`, включающей в себя значение десятичной дроби, основание системы счисления и статус (конечное представление или бесконечная десятичная дробь).

`has_finite_representation`:

- Вспомогательная функция, которая проверяет, имеет ли переданное вещественное число конечное представление в системе счисления с заданным основанием. Возвращает `true`, если представление конечное, и `false` в противном случае.

`check_inf_or_not`:

- Основная функция с переменным числом аргументов. Принимает переменное число аргументов типа `double`, представляющих десятичные дроби. Для каждой дроби определяет конечное или бесконечное представление в системе счисления с заданным основанием. Результат сохраняется в массиве структур `Numb`, который затем передается по указателю.

Функции позволяют анализировать конечность представления десятичных дробей в различных системах счисления и выводить информацию о них.

Задание 2.10 [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:

$$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_n x^n = g_0 + g_1(x-a) + \cdots + g_n(x-a)^n$$

Решение

Общая суть задачи:

Необходимо реализовать функцию с переменным числом аргументов для нахождения переразложения многочлена с заданными коэффициентами по степеням значения (x - a). Результат представляется в виде коллекции коэффициентов нового многочлена.

Основные функции:

`gorner_calculate`:

- Вспомогательная функция для вычисления значения многочлена методом Горнера. Принимает значение x, степень многочлена N и массив коэффициентов mas. Возвращает результат вычисления.

`derivative_of_polynomial`:

- Вспомогательная функция для вычисления производной многочлена. Принимает указатель на степень многочлена N и массив коэффициентов m. Модифицирует массив коэффициентов, умножая каждый коэффициент на свою степень.

`compute_coefficients`:

- Основная функция с переменным числом аргументов. Принимает точность epsilon, значение a, количество коэффициентов `count_coef` и сами коэффициенты многочлена. Вычисляет коэффициенты нового многочлена, представляющего переразложение исходного

многочлена по степеням (x - a). Результат записывается в динамически выделенный массив `result_coefficients`, который затем возвращается.

Функция позволяет анализировать переразложение многочлена по степеням (x - a) с использованием переменного числа аргументов и вычислять соответствующие коэффициенты.

Работа № 3.

Задание 3.1 [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 суть задачи:

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:

`addition`:

- Реализует сложение двух чисел без использования стандартных арифметических операций.

`check_base`:

- Проверяет, является ли указанное основание системы счисления степенью двойки. Если да, вычисляет маску и количество битов, необходимых для представления чисел в этой системе.

`reverse`:

- Инвертирует порядок символов в строке.

`number_to_crnt_base`:

- Основная функция для преобразования числа в систему счисления с указанным основанием. Выделяет память под строку и изменяет ее в процессе работы.

Данное решение [31] на Си реализует преобразование целых чисел из десятичной системы, в другую, через бинарные операции ^, & и ~. В коде также реализована обработка ошибок и очистка памяти.

Задание 3.2 [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:

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

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

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

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 суть задачи:

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

Основные функции:

`typedef Vector`:

- Определение структуры Vector.

`Norm_func`:

- Определение типа указателя на функцию для вычисления норм.

`infinity_norm`:

- Вычисление бесконечной нормы вектора.

`p_norm`:

- Вычисление p-нормы вектора.

`A_norm`:

- Вычисление нормы вектора в соответствии с матрицей A.

`printVector`:

- Вывод вектора на экран.

`free_vectors`:

- Освобождение памяти, выделенной под вектора.

`find_longest_vectors`:

- Основная функция, которая находит самые длинные векторы для каждой переданной нормы.

В представленном коде реализована программа на языке C, включающая структуры и функции для работы с векторами в n-мерном пространстве. Используется указатель на функцию и переменное число аргументов для нахождения самых длинных векторов для различных норм (бесконечной, p-нормы и нормы, определенной матрицей A). Программа демонстрирует применение указателей на функции и эффективное использование переменного числа аргументов

Задание 3.3 [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 суть задачи:

Программа читает информацию о сотрудниках из файла, сортирует их сначала по заработной плате, затем по фамилии, имени и id, а 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.

`sort_by_increase` и `sort_by_decrease`:

- Функции для сортировки массива структур в порядке возрастания и убывания соответственно.

`print_employees_to_file`:

- Функция для вывода отсортированных данных в выходной файл.

diff_file:

- Функция для сравнения путей к файлам.

validate_params:

- Функция для проверки и анализа параметров командной строки.

scan_eployee:

- Функция для считывания информации о сотруднике из файла.

Данная программа на языке C предназначена для сортировки информации о сотрудниках, считанной из файла, и вывода результатов в другой файл. Она использует динамические массивы и стандартную функцию qsort для эффективной сортировки данных в заданном порядке. Программа также предусматривает обработку ошибок и контроль памяти, что делает ее надежной и устойчивой.

Задание 3.4 [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 лексографическому компаратору)
   - отношения эквивалентности между двумя экземплярами типа String (лексографический 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 суть задачи:

   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:

Структура `Address`:

- Представляет информацию о получателе, включая город, улицу, номер дома, корпус, номер квартиры и индекс.

Структура `String`:

- Используется для удобной работы со строками переменной длины.

Структура `Mail`:

- Содержит информацию о почтовом отправлении, включая данные получателя, вес посылки, почтовый идентификатор, время создания, время вручения и др.

Структура `Post`:

- Представляет почтовое отделение, включая текущий адрес и массив почтовых отправлений.

free_mail:

- Освобождает память, выделенную для строковых данных в структуре `Mail`.

print_mail:

- Выводит информацию о почтовом отправлении в консоль.

`print_all_crnt`:

- Выводит все текущие посылки в консоль.

Функция `get_current_time`:

- Получает текущее системное время в формате строки.

Функция `compare_mail`:

- Сравнивает две почтовые отправки по индексу получателя и почтовому идентификатору.

Функция `compare_mail_for_3`:

- Сравнивает две почтовые отправки по времени создания.

Функция `add_mail_to_post`:

- Добавляет почтовую отправку в почтовое отделение и сортирует массив отправлений.

`search_mail`:

- Ищет почтовую отправку по её идентификатору.

`search_mails`:

- Ищет все доставленные и просроченные отправления.

`interactive`:

- Реализует интерактивный диалог с пользователем, предоставляя функционал по добавлению, поиску и удалению почтовых

отправлений, а также выводу информации о доставленных и просроченных отправлениях.

Программа представляет собой систему управления почтовыми отправлениями. Она использует структуры для хранения информации о получателе, почтовой отправке и почтовом отделении. Интерактивный диалог с пользователем позволяет добавлять, искать и удалять почтовые отправления, а также выводить информацию о доставленных и просроченных отправлениях. Программа обеспечивает удобный ввод данных, динамическое выделение памяти и обработку ошибок для надежной работы.

Задание 3.5 [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 суть задачи:

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:

`print_status`:

- Функция выводит сообщение об успешном выполнении операции или сообщение об ошибке в зависимости от переданного кода статуса.

`diff_file`:

- Функция сравнивает два файла по их путям и возвращает статус успеха, неудачи или ошибки в зависимости от результата.

`is_alpha_string` и `is_digits_only`:

- Функции проверяют, состоит ли строка только из букв или цифр соответственно.

`compare_by_id`, `compare_by_surname`, `compare_by_name`, `compare_by_group`:

- Функции сравнения для использования в функции сортировки qsort.

`free_students`:

- Функция освобождает память, выделенную под структуры студентов и их оценки.

`search_student`:

- Функция выполняет поиск студента в массиве по заданному критерию (id, фамилии, имени, группе) и возвращает статус успеха или неудачи.

`print_student_info`:

- Функция выводит информацию о студенте в файл.

`write_to_file`:

- Функция записывает информацию обо всех студентах в файл.

`calculate_average_exam_score`:

- Функция вычисляет средний балл за все экзамены для всех студентов.

`write_high_achievers_to_file`:

- Функция записывает в файл фамилии и имена студентов с баллом выше среднего.

`sort_students`:

- Функция сортирует студентов по заданному критерию (id, фамилии, имени, группе) с использованием qsort.

`user_interface`:

- Основная функция интерфейса пользователя, позволяющая выбирать различные действия (поиск, сортировка, запись в файл) через команды в консоли.

`read_students_from_file`:

- Функция считывает информацию о студентах из файла и создает массив структур типа `Student`.

Программа поддерживает обработку ошибок при вводе данных, выделении памяти, открытии файлов и других операциях, выводя соответствующие сообщения пользователю.

Задание 3.6 [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 суть задачи:

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:

`check_coordination_file_in` и `check_coordination_file`:

- Проверяют наличие автобусов в заданных координатах в одном маршруте и во всех маршрутах соответственно.

`read_and_convert_to_doub`:

- Считывает строку из файла и конвертирует её в значение типа double.

`read_check_line`:

- Считывает строку из файла, проверяет её на корректность и возвращает строку.

`add_in` и `add_out`:

- Добавляют новый узел с автобусом в маршрут (route) и основной список (main list) соответственно.

`parsing_input_data`:

- Читает данные из файла и заполняет структуры для представления маршрутов и автобусов.

`free_list_route` и `free_list_buses`:

- Освобождают память, выделенную под списки маршрутов и автобусов.

`check_coordination_crnt_bus` и `check_coordination`:

- Проверяют наличие координат, совпадающих у разных автобусов в одном маршруте и во всех маршрутах соответственно.

`check_time_crnt_bus` и `check_time_bus`:

- Проверяют совпадение времени прибытия и отправления в одном маршруте и во всех маршрутах соответственно.

`check_in_list` и `check_all`:

- Проверяют правильность порядка стартовых и финальных точек в одном маршруте и во всех маршрутах соответственно.

`count_route_in`:

- Подсчитывает количество стартовых и финальных точек в маршруте.

`bus_id_route`, `bus_id_length_track`, `max_route_length_bus_id`, `long_short_stop`, `stop_time`:

- Возвращают идентификаторы автобусов, удовлетворяющих определенным критериям.

`length_by_x_y` и `track_length_in`:

- Вычисляют расстояние между двумя точками и длину маршрута соответственно.

`longest_route`:

- Вычисляет длину самого длинного маршрута в маршрутном листе.

`interactive`:

- Запускает интерактивный режим взаимодействия с пользователем.

Каждая из этих функций выполняет конкретную задачу, необходимую для анализа данных о движении автобусов. Код охватывает широкий спектр проверок и вычислений, что делает его полезным для анализа и оптимизации системы маршрутов.

Задание 3.7 [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 суть задачи:

Программа представляет собой управление информацией о жителях поселения с использованием односвязного упорядоченного списка. Ввод данных осуществляется из файла, который содержит информацию о фамилии, имени, отчестве, дате рождения, поле и среднем доходе жителя. Для каждого жителя создается объект структуры `Liver`, а затем помещается в упорядоченный список по возрасту.

Основные функции:

`Liver`:

- Структура, представляющая информацию о жителе (фамилия, имя, отчество, дата рождения, пол, средний доход).

`Node`:

- Узел односвязного списка с информацией о жителе и указателем на следующий узел.

`Action`:

- Структура, представляющая действие (вставка, модификация, удаление) с указателем на узел и предыдущую информацию о жителе.

`Undo_stack`:

- Структура для хранения стека действий.

`init_undo_stack`:

- Инициализация стека отмены.

`free_undo_stack`:

- Освобождение памяти, занятой стеком отмены.

`push_action`:

- Добавление действия в стек отмены.

`Undo`:

- Отмена последних N/2 действий.

`print_status`:

- Вывод сообщения об ошибке по коду статуса.

`is_number`, `is_latin`, `is_valid_gender`, `is_in_range`:

- Проверка валидности различных типов данных.

`validate_data`:

- Проверка валидности данных жителя.

Данная программа на языке C представляет собой управление информацией о жителях поселения, используя структуры данных и файловый ввод/вывод. Система включает функционал поиска, изменения, удаления и добавления данных о жителях, а также поддерживает отмену последних модификаций. Программа обеспечивает валидацию вводимых данных и предоставляет пользователю удобный интерфейс в виде консольного меню.

Задание 3.8 [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 суть задачи:

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:

`split`:
- Разделяет строку на токены по заданным разделителям.

`get_file_ptr`:
- Получает указатель на файл по заданному пути.

`filter_and_read_line`:
- Считывает строку из файла, фильтрует комментарии и лишние символы.

`pre_build_Polynomial`:
- Предварительная сборка многочлена из строки, содержащей только один многочлен.

`free_array`:
- Освобождает память, выделенную под массив строк.

`build_Polynomial`:
- Собирает многочлен из строки, содержащей несколько многочленов.

`get_long_check`:
- Преобразует строку в целое число.

`validate_Polynomial`:
- Проверяет корректность записи многочлена.

`count_of_target`:
- Подсчитывает количество заданных символов в строке.

`validate_line`:
- Проверяет корректность строки с командой.

`run_command`:
- Выполняет операции над многочленами в соответствии с командой.

`task`:
- Основная функция, читающая команды из файла и выполняющая операции.

`print_user_prompt`:

- Выводит подсказку по использованию программы.

Общий принцип работы: программа считывает команды из файла, разделяет их на токены, затем выполняет операции над многочленами в соответствии с полученными командами. Результаты операций выводятся на экран.

Задание 3.9 [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 суть задачи:

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:

`is_file_empty`:
- Проверка, является ли файл пустым.

`read_word`:
- Считывание слова из файла, разделенного указанными сепараторами, в динамический буфер.

`add_to_nodes`:
- Добавление узла в массив узлов Nodes.

`build_tree_from_file`:
- Построение дерева поиска из файла.

`load_tree_from_file_r`:
- Вспомогательная функция для загрузки дерева из файла.

`load_tree_from_file`:
- Загрузка дерева из файла.

`clear_buffer`:
- Очистка буфера ввода.

`interactive`:
- Интерактивное взаимодействие с пользователем для выполнения дополнительных опций.

`tree_depth`:
- рекурсивно определяет глубину дерева.

`save_tree_to_file`:
- Сохранение дерева в файл.

`load_tree_from_file`:
- Загрузка дерева из файла.

`load_tree_from_file_r`:
- Вспомогательная функция для загрузки дерева из файла.

Реализовано приложение для анализа текста с использованием двоичного дерева поиска. Пользователь может 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 3.10 [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 суть задачи:

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

Основные функции:

`enum` status_codes:

- Определение перечисления для кодов статуса.

`print_status`:

- Функция вывода сообщений в зависимости от кода статуса.

`diff_file`:

- Сравнение двух файлов по их путям.

`struct Node`:

- Определение структуры узла дерева.

`free_node`:

- Освобождение памяти, занятой узлами дерева.

`new_node`:

- Создание нового узла.

`build_tree`:

- Построение дерева выражения.

`to_file`:

- Запись дерева в файл.

Код реализует построение дерева скобочного выражения для каждой строки входного файла и выводит их в наглядной форме в выходной файл. Он также предусматривает обработку ошибок, таких как проблемы с выделением памяти и ошибки при открытии файлов.

Работа № 4.

Задание 4.1 [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 суть задачи:

Данное приложение написано на языке программирования C и предназначено для организации макрозамен в тексте на основе директив #define.

Основные функции:

`diff_file`:
- Функция сравнивает два файловых пути и проверяет, совпадают ли они.

`free_table`:
- Освобождает память, выделенную под хеш-таблицу.

`create_hash_table`:
- Создает новую хеш-таблицу заданного размера.

`hash`:
- Вычисляет хеш-значение для заданного ключа с использованием основания 62.

`insert`:
- Вставляет новый элемент (ключ и значение) в хеш-таблицу.

`replace_substr`:
- Заменяет подстроки в строке другими подстроками.

`resize_hash_table`:
- Изменяет размер хеш-таблицы, когда необходимо перебалансировать длину цепочек.

`insert_with_balancing`:
- Вставляет элемент в хеш-таблицу и проверяет необходимость перебалансировки.

`parsing_file`:
- Обрабатывает содержимое текстового файла, выполняя замены последовательностей символов <def_name> на <value>, а также управляет хеш-таблицей.

Приложение читает текстовый файл, содержащий директивы #define, и обрабатывает его, выполняя макрозамены в остальном тексте с использованием хеш-таблицы для хранения соответствий имен и значений макросов.

Задание 4.2 [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 суть задачи:

Программа реализует простой интерпретатор операций над целочисленными массивами.

Основные функции:

`diff_file`:
- Сравнивает два файла по их путям.

`create_init_array`:
- Создает и инициализирует целочисленный массив.

`free_array`:
- Освобождает память, выделенную под массив.

`clear_all_arrays`:
- Освобождает память для всех массивов в хранилище.

`convert_str_to_long`:
- Преобразует строку в целое число.

`load_array_from_file`:
- Загружает данные из файла в массив.

`print_single_element`:
- Печатает элемент массива по указанному индексу.

`print_range`:
- Печатает элементы массива в заданном диапазоне.

`print_all`:
- Печатает все элементы массива.

`save_array_to_file`:
- Сохраняет массив в файл.

`fill_array_with_random`:
- Заполняет массив случайными числами.

`concatenate_arrays`:
- Конкатенирует два массива.

`remove_elements`:
- Удаляет элементы из массива начиная с указанного индекса.

`copy_elements`:

- Копирует элементы из одного массива в другой.

`compare_ascending`:

- Функция сравнения для сортировки по возрастанию.

`compare_descending`:

- Функция сравнения для сортировки по убыванию.

sort_array:

- Сортирует массив в указанном порядке.

`swap_elements`:

- Обменивает значения двух элементов.

`random_compare`:

- Сравнение для случайной сортировки.

`shuffle_array`:

- Перемешивает элементы массива.

`find_max`:

- Находит максимальное значение в массиве.

`find_min`:

- Находит минимальное значение в массиве.

`find_most_common`:

- Находит самый часто встречающийся элемент в массиве.

`find_average`:

- Находит среднее значение элементов массива.

`find_max_deviation`:

- Находит максимальное отклонение от среднего значения.

`print_stats`:

- Печатает статистику для массива (максимум, минимум, среднее и др.).

`parsing`:

- Парсит команды из файла и выполняет соответствующие операции.

Код содержит функции для работы с массивами, сортировки, поиска элементов и других операций. Каждая функция выполняет свою конкретную задачу в рамках общей логики интерпретатора.

Задание 4.5 [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 суть задачи:

Программа представляет собой консольное приложение, которое принимает в качестве аргументов командной строки пути к текстовым файлам. Каждый файл содержит арифметические выражения, разделенные новой строкой. Программа выполняет несколько шагов для каждого

выражения: проверка баланса скобок в выражении, построение обратной польской записи для выражения, вычисление значения 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 обнаружены ошибки.

Основные функции:

`print_status_codes`:
- Вывод информации о статусе выполнения операции в стандартный поток вывода.

`print_info`:
- Вывод информации об арифметическом выражении, его результате и обратной польской записи.

`bpow`:
- Возведение числа в степень.

`is_operator`:
- Проверка, является ли символ оператором.

`get_priority`:
- Получение приоритета оператора.

`to_postfix_notation`:
- Преобразование инфиксного выражения в обратную польскую запись.

`solution`:
- Вычисление значения выражения из обратной польской записи.

`check_files`:
- Проверка файлов на наличие ошибок в выражениях и их обработка.

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 4.6 [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 суть задачи:

Программа представляет собой консольное приложение, которое строит таблицу истинности для заданной булевой формулы. Булева формула представлена в виде инфиксной записи, затем преобразуется в постфиксную запись, по которой строится дерево выражения. Для каждой комбинации значений переменных из формулы вычисляется результат логического выражения.

Основные функции:

`operator`:
- Проверка, является ли символ оператором.

`print_var`:
- Вывод переменных в заголовок таблицы.

`print_infix`:
- Вывод инфиксной формы выражения в файл.

`print_header`:
- Вывод заголовка таблицы.

priority:
- Получение приоритета оператора.

`is_two_char_operator`:
- Проверка, является ли оператор двухсимвольным.

`repeat_check`:
- Проверка повторяющихся переменных в формуле.

`count_var`:
- Подсчет уникальных переменных в формуле.

`create_filename`:
- Генерация случайного имени файла.

`solution_from_tree`:
- Вычисление значения выражения из дерева.

`build_truth_table`:
- Построение таблицы истинности для булевой формулы.

`infix_to_postfix`:
- Преобразование инфиксной формы в постфиксную.

`build_tree`:
- Построение дерева выражения из постфиксной формы.

`solution`:
- Основная функция решения задачи, вызывающая другие функции

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 4.7 [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.

Решение

Общая суть задачи:

Этот код представляет собой простой интерпретатор, способный выполнять базовые арифметические операции и операции присваивания, а также выводить значения переменных.

Основные функции:

`cells_arr_init`:
- Инициализирует массив Memory_cell.

`free_memory`:
- Освобождает выделенную память для массива Memory_cell.

`compare_memory_cells`:
- Сравнивает две ячейки памяти по именам переменных.

`find_memory_cell`:
- Ищет переменную в массиве Memory_cell с помощью бинарного поиска.

`insert`:
- Вставляет новую переменную в массив Memory_cell, сортируя его при необходимости.

`perform_operation`:
- Выполняет операцию над двумя значениями с проверкой на переполнение и возвращает результат.

`parse_operation`:
- Определяет тип операции на основе символа.

`is_operator`:
- Проверяет, является ли символ оператором.

`parse_long`:
- Преобразует строку в целое число с проверкой ошибок.

`parse_instruction_parts`:
- Разбирает отдельные части инструкции.

`parse_instruction`:

- Парсит инструкции из файла, выполняет операции и выводит результаты.

Этот код позволяет читать инструкции из файла и выполнять их, управляя переменными и выводя результаты операций.

Заключение

Этот курсовой проект был захватывающим и полезным, в ходе которого я изучил и применил множество важных концепций и технологий в области программирования. Вот несколько ключевых моментов, которые я выделяю после завершения этой работы:

Изучение языка программирования C:

- Проект включал в себя написание кода на языке C. Это позволило мне улучшить свои навыки в этом языке и лучше понять его особенность.

Структуры данных и алгоритмы:

- Работа с различными структурами данных, такими как списки, стеки и деревья и т.д., позволила мне углубить свои знания в области алгоритмов и эффективного управления данными.

Управление памятью:

- Правильное решение задач требовало внимания к управлению памятью, включая выделение и освобождение динамической памяти. Это дало мне опыт работы с динамическим выделением памяти и обработкой ошибок.

Взаимодействие с файлами:

- Работа с файлами и чтение инструкций из текстового файла позволило мне освоить техники обработки файлов в языке C.

Обработка ошибок:

- Весь процесс решений задач требовал внимания к обработке ошибок, что подняло мой опыт работы с обработкой ошибок и созданием более надежных программ.

Практический опыт в программировании:

- Вся работа предоставила мне практический опыт в различных аспектах программирования, начиная от проектирования структур данных до реализации алгоритмов.

Улучшение навыков отладки и тестирования:

- В ходе разработки приложений я улучшил свои навыки отладки и тестирования, что сделало код более стабильным и надежным.

Общий вывод: Курсовая работа предоставила мне ценный опыт в различных областях программирования, от языка C до работы с файлами и обработки ошибок. Я улучшил свои навыки в разработке программ, повысил

понимание структур данных и алгоритмов, а также расширил свой арсенал техник отладки и тестирования. Этот проект стал важным этапом в моем обучении, и я уверен, что полученные знания и опыт будут полезными в моем дальнейшем пути как программиста.

Список использованных источников

Алгоритмика

Викиконспекты

Prog-cpp.ru

MAXimal :: algo

Практические занятия и лекции по Математическому практикуму

Приложение

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab1/lab1_1 [1]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab1/lab1_2 [2]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab1/lab1_3 [3]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab1/lab1_4 [4]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab1/lab1_5 [5]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab1/lab1_6 [6]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab1/lab1_7 [7]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab1/lab1_8 [8]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab1/lab1_9 [9]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab1/lab1_10 [10]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab2/lab2_1 [11]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab2/lab2_2 [12]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab2/lab2_3 [13]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab2/lab2_4 [14]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab2/lab2_5 [15]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab2/lab2_6 [16]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab2/lab2_7 [17]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab2/lab2_8 [18]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab2/lab2_9 [19]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab2/lab2_10 [20]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab3/lab3_1 [21]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab3/lab3_2 [22]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab3/lab3_3 [23]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab3/lab3_4 [24]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab3/lab3_5 [25]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab3/lab3_6 [26]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab3/lab3_7 [27]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab3/lab3_8 [28]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab3/lab3_9 [29]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab3/lab3_10 [30]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab4/lab4_1 [31]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab4/lab4_2 [32]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab4/lab4_5 [33]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab4/lab4_6 [34]

https://github.com/MaximusPokeZ/Math_workshop_by_MZ/tree/main/Labs/lab4/lab4_7 [35]