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: Доянова В.А.

Группа: М8О-211Б-22

Преподаватель: Романенков А.М.

Оценка:

Дата:

Москва, 2023

Содержание

Описание задания 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.

Выполнение задания подразумевает:

- реализацию задания на языке программирования C (стандарт C99 и выше)
- описание реализованного решения (текстовое описание решения с точки зрения логики работы решения, использованных алгоритмов и структур данных)

Основная часть

Список заданий №1

Задание №1. Обработка чисел

Условие задания: 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: в первую очередь осуществляется проверка на количество введенных аргументов, валидность числа и флага. Проверка числа осуществляется с помощью функции «number_check», которая проходит по числу и проверяет, чтобы в его записи не было лишних символов. Проверка флага осуществляется с помощью функции «flag_check», где проверяется принадлежность флага к списку допустимых флагов. Обе функции возвращают 1 в случае успеха и 0 в противном случае.

Далее, в случае успешного прохождения проверок, происходит обработка числа в зависимости от введенного флага.

Флаг «h». Обработка осуществляется с помощью функции «flag_h». Если число не входит в диапазон от 0 до 100, то функция выводит ошибку. Иначе с помощью цикла «for» происходит поиск всех чисел, кратных введенному. Если найдено хотя бы одно число, то переменная «status» становится равной 1. Если после цикла она равняется 0, то выводится сообщение о том, что соответствующих чисел не было найдено.

Флаг «p». Обработка осуществляется с помощью функции «flag_p». Если число равняется одному, то функция возвращает ноль, а далее выводится сообщение о том, что число простое. Если число меньше одного, то функция возвращает -1, затем выводится сообщение о том, что введенное число должно быть больше нуля. Если число больше или равно двум, то для проверки его на простоту используется цикл «for», который идет от значения 2 и до самого числа. Если в процессе поиска было найдено число, которое кратно введенному, то возвращается 0, что означает, что число не простое. Иначе функция возвращает 1.

Флаг «s». Функция «flag_s» обрабатывает введенное число в строковом представлении. В ней используется счетчик «position», который отвечает за рассматриваемую цифру в числе. Если число отрицательное (то есть первый символ слева это знак минуса), то он пропускается. Если далее следуют ведущие нули, то они также пропускаются с помощью цикла «while». Далее в цикле «for», начиная с текущего значения «position» и до значения длины числа, происходит печать всех цифр через пробел.

Флаг «e». С помощью функции «flag_e» происходит печать таблицы степеней. Если введенное число не находится в диапазоне от 1 до 10, то выводится ошибка о неверном числе. Иначе сначала выводится верхняя часть таблицы, а затем с помощью двух циклов «for» происходит вывод значений степеней. Первый цикл выводит текущую степень. Второй находит значение числа в этой степени с помощью вспомогательной функции «calculations_for_e», куда отправляется число и текущая степень. В этой функции изначально переменная «total» (типа «long long int» для того, чтобы поместились самые большие по значению результаты) равняется введенному числу. Затем в цикле «for» число умножается на «total» пока значение «i» не дойдет до нужной степени. Затем значение «total» возвращается обратно, где печатается в таблице.

Флаг «a». Обработка осуществляется с помощью функции «flag_a». В неё передается введенное число, в самой функции определена переменная «result», которая и будет результатом суммирования. Если число положительное, то для определения суммы используется следующая формула [1]: $S = \frac{n(n+1)}{2}$, где «S» это сумма, а «n» - введенное число. Если число отрицательное или равняется нулю, то функция вернет ноль.

Флаг «f». Обработка осуществляется с помощью функции «flag_f». Если число больше 20 или меньше 0, то функция возвращает 0, а далее

выводится сообщение о неподходящем числе. Функция обрабатывает числа до 20, так как иначе значение факториала перестанет правильно рассчитываться из-за слишком большого значения числа, оно перестанет помещаться в используемый тип данных («long long int»). Если число равняется нулю, то функция возвращает 1 – значение факториала для нуля. Иначе с помощью цикла «for» происходит подсчет факториала: цикл идет от значения 2 до числа включительно, каждую итерацию умножая «total» (изначально равняется 1) на текущее значение «i», так как факториал представляет из себя произведение всех натуральных чисел от 1 до «n» включительно. После завершения цикла функция возвращает значение «total».

Задание №2. Вычисление e, π, $ln\,2$, $\sqrt{2}$, γ

Условие: 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	$\pi = \lim\limits_{n\to\infty}\frac{(2^n n!)^4}{n((2n)!)^2}$	$\pi = 4\sum\limits_{n=1}^{\infty}\frac{(-1)^{n-1}}{2n-1}$	$\cos x = -1$
ln 2	$ln\,2 = \lim\limits_{n\to\infty} n\left(2^{\frac{1}{n}}-1\right)$	$ln\,2 = \sum\limits_{n=1}^{\infty}\frac{(-1)^{n-1}}{n}$	$e^x = 2$
$\sqrt{2}$	$\sqrt{2} = \lim\limits_{n\to\infty}(x_n)$, 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	$\gamma = \lim\limits_{m\to\infty}\left(\sum\limits_{k=1}^{m} C_m^k \frac{(-1)^k}{k}\ln(k!)\right)$	$\gamma = -\frac{\pi^2}{6} + \sum\limits_{n=1}^{\infty}\left(\frac{1}{\lvert\sqrt{k}\rvert^2} - \frac{1}{k}\right)$	$e^{-x} = \lim\limits_{t\to\infty}\left(\ln t \prod\limits_{p\le t,\ p\in P}\frac{p-1}{p}\right)$

Точность вычислений (значение эпсилон) подаётся программе в качестве аргумента командной строки. Продемонстрируйте выполнение реализованных функций.

Решение: первоначально проверяется количество введенных аргументов (должен быть введен один аргумент – точность вычислений). Далее происходит проверка введенного числа с помощью функции «number_check». Если число имеет лишние знаки в своей записи, то функция вернет 0, выведется сообщение об ошибке и программа завершится.

Если введено правильное число, то оно используется для инициализации переменной «eps» (эпсилона): переменная будет равна 10 в степени, равной введенному числу, умноженному на -1. Далее происходит вычисление значений.

Вычисление пределов.

Число *e.* Значение предела для этого числа находится в функции «ex_lim». В переменную «result_1» записывается функция из предела, представленного в таблице, а в переменную «result_2» записывается эта же функция, но с «n», увеличенным на единицу. Затем, в цикле «while» (пока модуль разности «result_2» и «result_1» больше эпсилона, так как мы используем тип данных «double») «n» каждый раз увеличивается на единицу, а полученный результат сохраняется. Таким образом находится предел данной последовательности.

Число π. Значение предела для этого числа находится в функции «p_lim». В переменную «result_1» изначально записывается 0, а в переменную «result_2» записывается 4 (так как значение «n» увеличено на единицу). Затем в цикле «while» «n» каждый раз увеличивается на единицу, а «result_2» умножается на значение функции (вместо изначальной функции мы берем вариант, разложенный с помощью рекуррентной формулы [2], чтобы упростить вычисления) с обновленным «n». Полученное значение сохраняется в «result_1». Таким образом находится предел, и результат возвращается.

Число $ln\,2$. Значение предела для этого числа находится в функции «ln_lim». В переменную «result_1» изначально записывается значение функции, а в переменную «result_2» записывается значение функции с «n», увеличенной на единицу. Затем в цикле «while» «n» каждый раз увеличивается на единицу, а «result_2» приравнивается к значению функции с обновленным «n». Полученное значение сохраняется в «result_1». Таким образом находится предел, и результат возвращается.

Число $\sqrt{2}$. Значение предела для этого числа находится в функции «sqrt_lim». В переменную «result_1» изначально записывается 0, а в переменную «result_2» записывается -0.5, так как это дано по условию. Затем в цикле «while» «result_2» приравнивается к значению функции $x_n - \frac{x_n^2}{2} + 1$, где вместо x_n используется «result_1». Полученное значение сохраняется в «result_1». Таким образом находится предел, и результат возвращается.

Число γ. Значение предела для этого числа находится в функции «gamma_lim». В переменные «result_1» и «result_2» изначально записывается 0. Затем начинается цикл «do-while». Переменные «sum» и «fact» каждый раз равняются нулю и единице перед внутренним циклом «for». В данном цикле считается значение очередного элемента суммы, пока «i» не дойдет до значения «m» (оно увеличивается после данного цикла на единицу). Для получения значения элемента используется дополнительная функция «C», в которой считается число сочетаний по формуле $C_m^k = \frac{m!}{(m-k)!k!}$. Соответственно, для этого используется функция «fac». После получения суммы с помощью цикла «for» «result_2» приравнивается к полученному значению. Таким образом находится предел, и результат возвращается.

Вычисление ряда/произведения.

Значение ряда для числа *e* находится в функции «ex_row». В этой функции используется цикл «do-while» (пока модуль текущего элемента больше эпсилона), в теле которого находится вычисление значения функции $\frac{1}{n!}$, увеличение «n» каждый раз на единицу. Значение факториала находится с помощью вспомогательной функции «fac», она работает рекурсивно: если число равняется нулю, то возвращается единица, иначе происходит умножение числа на факториал предыдущего [3]. Таким образом, полученное значение функции прибавляется к сумме, которое в конце работы функции возвращается.

Значение ряда для числа π находится в функции «p_row». В этой функции используется цикл «do-while», в теле которого находится вычисление значения функции $\frac{(-1)^{n-1}}{2n-1}$, увеличение «n» каждый раз на единицу. Таким образом, полученное значение функции прибавляется к сумме, которое в конце работы функции возвращается, умноженное на 4.

Значение ряда для числа $ln\,2$ находится в функции «ln_row». В этой функции также используется цикл «do-while, в теле которого находится вычисление значения функции $\frac{(-1)^{n-1}}{n}$, увеличение «n» каждый раз на единицу. Таким образом, полученное значение функции прибавляется к сумме, которое в конце работы функции возвращается.

Значение ряда для числа $\sqrt{2}$ находится в функции «sqrt_row». В этой функции перед использованием цикла инициализируется переменная «power», которая изначально равна 0.25, так как «k» в функции изначально равняется двум. Далее используется цикл «do-while», в теле которого находится вычисление значения функции $2^{2^{-k}}$, но вместо суммирования происходит умножение полученного элемента на результат, а также умножение степени на 0.5. Таким образом, полученное произведение в конце работы функции возвращается.

Значение ряда для числа γ находится в функции «gamma_row». В этой функции используется цикл «do-while», в теле которого находится вычисление значения функции $\frac{1}{|\sqrt{k}|^2} - \frac{1}{k}$, увеличение «k» каждый раз на единицу. Также, так как в знаменателе содержится модуль, то учитываются только элементы, большие эпсилона. Таким образом, полученное значение функции прибавляется к сумме, а в конце работы функции возвращается разность полученной суммы и $\frac{\pi^2}{6}$.

Решение уравнения.

Результат уравнения для числа *e* находится в функции «ex_newton». Вычисление происходит с помощью метода Ньютона [4]. В переменные «a» и «b» записываются границы отрезка, на котором будет происходить поиск. В «result» изначально записывается приближенное значение корня, в переменную «x» записывается разность результата и функции, деленной на её производную. Далее в цикле «while» значение «result» приравнивается к «x», а вторая переменная каждый раз обновляет своё значение в зависимости от значения «result». После работы цикла функция возвращает значение «x».

Результат уравнения для числа π находится в функции «p_newton», для числа $ln\,2$ в функции «ln_newton», для числа $\sqrt{2}$ в функции «sqrt_newton». Вычисление также происходит с помощью метода Ньютона [4], то есть в функции всё происходит аналогично функции «ex_newton».

Результат уравнения для числа γ находится в функции «gamma_newton». Так как в функции используется произведение, а также предел, то для вычисления используются дополнительные функции «gam_lim», «multip». Функция «gam_lim» аналогична предыдущим функциям, где находился предел. В ней используется функция «multip», которая умножает логарифм на дробь. В дроби используется «p», которое по условию должно быть простым числом. Проверка на это происходит в функции «prime», где, если число равняется 0 или 1, то возвращается 0 (значит число не является простым), если число равняется 2, то возвращается 1 (значит число простое), а иначе в цикле «for» проверяется, имеет ли число делители. Таким образом, после окончания цикла «do-while», в теле которого каждый раз находится значение функции и прибавляется к переменной «result_2», возвращается значение «result_2».

Ниже представлен результат работы программы:

```
PS C:\Users\User\Desktop\Math_Prac\1_2> .\a.exe 3
e limit: 2,68146442
e row: 2,71825397
e function: 2,71828183
____________________
p limit: 3,16976668
p row: 3,14358866
p function: 3,14228266
____________________
ln2 limit: 0,70747218
ln2 row: 0,69264743
ln2 function: 0,50050032
____________________
sqrt limit: 1,41371798
sqrt row: 1,41230029
sqrt function: 1,41421356
____________________
gamma limit: 0,56367838
gamma row: 0,57775815
gamma function: 0,58313576
```

Рисунок 1. Результат работы программы №2

Задание №3. Обработка набора чисел

Условие: 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: первоначально происходит проверка флага на правильность ввода. Это осуществляется с помощью функции «flag_check», в которой с помощью «strcmp» проверяется длина введенного флага, а далее совпадение его с каким-либо флагом из списка допустимых флагов. В случае успеха возвращается 1, иначе 0.

Флаг «q». До начала работы функции происходит проверка на количество введенных аргументов (оно должно равняться 6 для данного флага), а также проверка введенных чисел на корректность с помощью функции «number_check». Введенные коэффициенты помещаются в массив «numbers», который сортируется с помощью функции «sort_numbers». В данном функции с помощью дополнительных функций «find_min» «find_max», куда также передаются коэффициенты, находится максимум и минимум среди введенных чисел (это осуществляется с помощью обычного сравнения чисел между собой), а затем числа в массиве меняются местами так, чтобы массив был отсортирован.

После сортировки чисел начинается цикл «do-while», который работает, пока есть доступные комбинации коэффициентов. Комбинации проверяются с помощью функции «combination», где числа в массиве меняются местами, избегая повторений с помощью учета индексов чисел. В основной функции «flag_q», которая вызывается в цикле, первоначально рассчитывается дискриминант квадратного уравнения, сохраняемый в переменной «discr», а затем, если дискриминант больше эпсилона, то находятся корни уравнения и выводятся на экран. Если дискриминант равен нулю, то корень один, и он также выводится на экран. Иначе корней нет, выводится соответствующее сообщение.

Флаг «m». Сначала происходит проверка на количество введенных аргументов. Если количество правильное, то происходит проверка введенных чисел с помощью функции «number_check». В случае успешной проверки вызывается функция «flag_m». В этой функции, если одно или оба числа равняются 0, то возвращается 2, которое указывает на данный случай,

затем выводится соответствующее сообщение. Если числа делятся друг на друга без остатка, то возвращается 1 (значит одно число кратно другому). Иначе возвращается 0 (одно число не кратно другому).

Флаг «t». Сначала проверяется количество введенных аргументов. Если оно верно, то происходит проверка чисел на валидность. Далее вызывается функция «flag_t». Сначала в функции находится максимальное и минимальное из введенных чисел. Далее, если все числа больше эпсилона, то происходит проверка с помощью теоремы Пифагора (из большей стороны вычитается сумма оставшихся, все длины во второй степени). Если проверка пройдена, то возвращается 1, иначе возвращается 0.

Задание №4. Обработка файла

Условие: 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: первоначально проверяется количество введенных аргументов. Если оно правильное, то имя входного файла помещается в переменную «input».

Далее, если количество аргументов равно 4, значит, был введено еще имя файла, куда нужно записать вывод программы. Поэтому нужно проверить флаг на то, чтобы он был из списка флагов, которые предполагают ввод файла от пользователя. Если всё правильно, то

возвращается 1. Иначе, если флаг не из списка в принципе, то возвращается 0, а если флаг не предполагает ввода имени файла от пользователя, то возвращается 2.

Если количество аргументов равно 3, то проводится аналогичная проверка флага. Эти проверки осуществляются в функции «flag_check», где происходит сравнение введенной строки с существующими флагами. Если проверка пройдена, то генерируется имя файла с помощью функции «output_name», куда передается имя входного файла, длина этого имени, длина окончания (окончание – «out_»), а также адрес переменной «output», куда будет записано итоговое имя файла с выводом.

В функции «output_name» сначала ищется позиция, где мы встретим символ «\». Для этого в цикле «for» осуществляется проход с конца имени файла. Это нужно для случая, когда имя файла подаётся не просто как «input.txt», а как полный путь к файлу. В переменную «position» записывается найденная позиция. Далее выделяется память для имени, размер которой будет равен сумме длины имени и длины окончания. Если «position» равняется нулю, то осуществляется копирование в имя сначала окончания, а затем имени файла. Иначе выделяется память для переменной «end_input», в которую перенесется путь к файлу (это случай, когда имя файла подаётся не просто как «input.txt», а как полный путь). В цикле «for» происходит копирование пути. Далее в итоговое имя сначала копируется имя файла до «position», затем копируется окончание, а затем оставшаяся часть имени. Далее переменная «end_input» очищается.

После прохождения всех проверок и создания имени выходного файла (если это требуется) файлы открываются для чтения и записи. Затем осуществляется обработка, согласно введенному флагу.

Флаг «d». Обработка осуществляется с помощью функции «flag_d», куда передается входной и выходной файл. В цикле «while», пока не был достигнут конец входного файла, происходит посимвольная обработка записей. Если очередной символ является цифрой или концом файла, то этот символ пропускается. Иначе символ записывается в выходной файл.

Флаг «i». Обработка осуществляется с помощью функции «flag_i», куда передается входной и выходной файл. В данной функции также используется цикл «while» до момента, пока не будет достигнут конец входного файла. В цикле происходит посимвольная обработка файла. Если очередной символ не является переносом строки или концом файла, а также является латинской буквой, то счетчик «count» увеличивается на единицу. Если символ – перенос строки или конец файла, то в выходной файл

записывается строка с полученным количеством букв в обработанной строке (значение «count»), а затем соответствующая переменная обнуляется.

Флаг «s». Обработка осуществляется в функции «flag_s». Функция работает аналогично функции «flag_i», изменение состоит в том, что счетчик увеличивается, когда текущий символ не является латинской буквой, цифрой и пробелом.

Флаг «a». Обработка осуществляется с помощью функции «flag_a». Работа функции похожа на функцию «flag_d». Отличие в том, что, если символ не является цифрой, то в выходной файл он записывается с помощью спецификатора типа «%X», который позволяет записать шестнадцатеричное число в файл. Символы цифр записываются в выходной файл без изменений.

После обработки флага происходит очищение переменной «output», а также закрытие открытых файлов.

Задание №5. Вычисление значения сумм

Условие: 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: первоначально проверяется количество введенных аргументов. Если оно правильное, то осуществляется проверка введенных чисел и эпсилона с помощью функций «number_check» и «eps_check». В этих функциях в цикле происходит посимвольная проверка чисел в строковом представлении. Если в записи чисел содержится больше одного знака «.», лишние символы, то возвращается 0, что означает неправильно введенное число.

Нахождение значения суммы «a» происходит в функции «function_a». В цикле «do-while» (работает, пока модуль текущего элемента больше эпсилона) происходит нахождение очередного элемента согласно функции $\frac{x^n}{n!}$. Для этого используется вспомогательная функция «fac», работа которой была описана ранее, а также функция «pow» для возведения в степень.

Найденный элемент прибавляется к сумме, «n» увеличивается на единицу. После завершения цикла сумма возвращается.

Нахождение значения суммы «b» происходит в функции «function_b» и происходит аналогично функции «function_a». Отличие состоит в том, что в «function_b» очередной элемент равен значению функции $\frac{(-1)^n x^{2n}}{(2n)!}$.

Нахождение значения суммы «c» происходит в функции «function_c». Отличие от функций «function_a» и «function_b» состоит в том, что здесь вычисление происходит не самой функции в том виде, в котором она дана по условию, а функция имеет вид, полученный после разложения с помощью рекуррентной формулы [2]. Сумма имеет изначальное значение 1, так как это результат функции при «n» равному 0. Таким образом, значение элемента каждый раз умножается на новое, а затем прибавляется к сумме, которая после завершения цикла возвращается.

Нахождение суммы «d» происходит в функции «function_d», которая работает аналогично функции «function_c». Отличие состоит в изначальном значении суммы и элемента (при «n» равному 1), а также в вычислении очередного элемента. Эти изменения связаны с функцией, которая дана в условии, логика нахождения суммы осталась та же.

Также в функциях «function_c» и «function_d» есть ограничение на значение вводимого «x», так как эти суммы сходятся только при «x», принадлежащему диапазону от -1 до 1 (не включительно). Если на вход подается значение, не принадлежащее диапазону, то функции возвращают 0, иначе они возвращают результат.

```
PS C:\Users\User\Desktop\Math_Prac\1_5> .\a.exe 0.00001 0.5
Result of a: 1.6487211682
Result of b: 0.8775825622
Result of c: 2.9004840312
Result of d: -0.1055722372
```

Рисунок 2. Результат работы программы №5

Задание №6. Вычисление значения интегралов

Условие: 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. $\int_0^1 -\frac{x^2}{2} dx$

c. $\int_0^1 ln\frac{1}{1-x} dx$

d. $\int_0^1 x^x dx$

Решение: для отслеживания ошибок используется тип данных перечисление, в котором определены возможные ошибки, которые могут возникнуть в процессе обработки. В программе объявлено перечисление с именем «errors», где определены два значения: «wrong_amount» и «wrong_eps».

Первоначально происходит проверка на количество введенных аргументов. Если оно неправильное, то выводится сообщение об ошибке и программа возвращает «wrong_amount». Далее происходит проверка введенного эпсилона. Если проверка неуспешная, то программа возвращает «wrong_eps». После этого происходит вычисление интегралов.

Вычисление интеграла «a» происходит в функции «integral_a». Вычисление происходит с помощью метода парабол (Симпсона) [5]. В переменные «a» и «b» записываются границы интеграла – 0 и 1 соответственно. Количество отрезков «h» это разность правой и левой границы, умноженной на введенный пользователем эпсилон. В значение «x» изначально записывается сумма левой границы и количества отрезков. Дальше начинается цикл «while», который работает, пока «x» меньше «b». В цикле происходит вычисление по формуле. Функция, которая представлена в интеграле, вычисляется с помощью функции «function_a» (функция возвращается посчитанное значение для текущего переданного «x»). Таким образом, из функции «integral_a» возвращается посчитанное с помощью метода Симпсона значение интеграла.

Интегралы «b», «c» и «d» вычисляются в функциях «integral_b», «integral_c», «integral_d». В них используются вспомогательные функции «function_b», «function_c», «function_d» для получения результата функции при определенном «x». Принцип работы всех функций аналогичен функциям «integral_a» и «function_a».

Все результаты выводятся на экран после вычисления.

```
PS C:\Users\User\Desktop\Math_Prac\1_6> .\a.exe 0.00001
function a answer: 0.822468
function b answer: 0.855628
function c answer: 1.000133
function d answer: 0.783437
```

Рисунок 3. Результат работы программы №6

Задание №7. Обработка файлов

Условие: 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 из 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: для отслеживания ошибок используется тип данных перечисление, в котором определены возможные ошибки, которые могут возникнуть в процессе обработки. В программе объявлено перечисление с именем «errors», где определены значения: «success» (успешное завершение), «wrong_amount» (неверное количество аргументов),

«wrong_flag» (неверный флаг), «not_open» (файл не был открыт) и «memory_error» (ошибка выделения памяти).

Первоначально проверяется количество введенных аргументов, далее проверяется введенный флаг с помощью функции «flag_check», которая аналогична функции проверки флага из предыдущих описанных заданий.

После успешных проверок начинается обработка флага.

Флаг «-r». Первоначально происходит проверка количества введенных аргументов, так как на вход должно быть два файла с лексемами и файл для записи результата. Если количество верное, а также все файлы открыты, то вызывается функция «flag_r», куда передаются все файлы.

Сначала в этой функции получается длина первого и второго файла с записями. Это происходит с помощью функции «get_file_length», куда передается файл. В данной функции с помощью «fseek» указатель перемещается на конец файла. Далее в переменную «length» записывается значение указателя текущего положения в файле, определенное с помощью «ftell». После этого указатель перемещается обратно на начало файла, а полученная длина возвращается.

После получения длины файлов используется функция «get_file», которая позволяет получить лексемы из файла. Для этого инициализируются пустые строки «list_1» и «list_2», которые передаются в функцию вместе с файлом и его длиной. В данной функции выделяется память для строки, а дальше в цикле «while», который работает до конца файла, происходит посимвольная обработка файла. Если очередной символ является буквой или цифрой, то он записывается в строку. Если символ – это пробел, а также цикл не дошел до конца файла, то в строку записывается пробел, который разделяет лексемы.

После получения строк с лексемами начинается запись итогового файла. Если длина первого файла больше, чем второго, то запускается цикл «for», который идет до длины первого файла. В этом цикле сначала записывается лексема из первого файла (пока очередной символ не является пробелом), а далее лексема из второго. Если программа дошла до конца второго файла, то вместо лексем из него в выходной файл записываются пробелы. Далее строки очищаются. Если длина второго файла больше, то цикл работает аналогично, только его границей является длина второго файла (то есть первый и второй файл меняются местами).

После завершения функции «flag_r» файлы закрываются.

Флаг «-a». Первоначально проверяется количество поданных на вход аргументов, так как пользователь должен ввести файл с записями и файл для записи. Если проверка пройдена, а также файлы открыты, то вызывается функция «flag_a».

Сначала в функции получается длина входного файла с помощью функции «get_file_length». Далее также получаются лексемы из файла с помощью функции «get_file». После этого начинается цикл «for», который длится до момента, пока «i» меньше длины файла. Первая лексема записывается в файл. Далее, если встретился пробел, то счетчик «count», который подсчитывает лексемы, увеличивается на единицу, а пробел также записывается в файл. После этого, если счетчик делится на 2 без остатка (и не делится на 10, по условию), то лексема в файл записывается со строчными символами вместо заглавных (символы меняются с помощью «tolower»). Если лексема десятая, то сначала все символы букв становятся строчными, а затем в файл записывается ASCII-код символа, переведенный в систему счисления с основанием 4. Перевод осуществляется с помощью функции «to_base_4», куда передается ASCII-код.

В этой функции используется цикл «while», который длится, пока число не равняется нулю. В цикле к промежуточному результату «answer» каждый раз прибавляется остаток от деления поданного числа на 4. Затем число делится на 4, а «answer» умножается на 10. То есть происходит перевод числа в необходимую систему счисления. После завершения первого цикла результат переворачивается в другом цикле (так как изначально получившийся результат записан в перевернутом виде). В этом цикле «total» (итоговый результат), умноженное на 10, суммируется с остатком от деления промежуточного результата на 10. Затем «answer» делится на 10. Таким образом получается последняя цифра «answer», которая затем отбрасывается. Цикл продолжается, пока промежуточный результат не равен 0. Затем «total» возвращается из функции.

Если лексема пятая, то в цикле «while» получается символ лексемы, который помещается в переменную «ascii» (то есть получается ASCII-код символа). Затем код записывается в файл с помощью спецификатора типа «%o», который переводит число в систему счисления с основанием 8.

После завершения работы цикла строка, используемая для хранения лексем, очищается. После завершения работы функции используемые файлы закрываются.

Задание №8. Определение системы счисления

Условие: 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: для отслеживания ошибок используется тип данных перечисление, в котором определены возможные ошибки, которые могут возникнуть в процессе обработки. В программе объявлено перечисление с именем «errors», где определены значения: «success» (успешное завершение), «wrong_amount» (неверное количество аргументов), «wrong_flag» (неверный флаг), «not_open» (файл не был открыт) и «memory_error» (ошибка выделения памяти).

Первоначально проверяется количество введенных аргументов. Если количество аргументов верное, а также файлы были успешно открыты, то вызывается функция «function», куда передаются файлы.

В данной функции сначала получается размер входного файла, а также его лексемы с помощью функций «get_file_length» и «get_file», аналогичных функциям из предыдущих заданий. После успешного получения также выделяется память для строки «number», куда будут записываться полученные числа. Затем начинается цикл «for», который длится, пока «i» меньше длины файла. В цикле обрабатываются числа из файла. Для каждой лексемы необходимо определить систему счисления. Это происходит переводом символа в его численное значение. Если очередной символ – цифра, то перевод осуществляется с помощью вычитания из символа `0` и прибавления единицы. Если символ это буква,

то сначала она становится строчной (с помощью «tolower»), а затем из полученного символа вычитается `a` и прибавляется 11 (таким образом, если у нас символ «A», то, после превращения в строчный символ у нас получится 97-97 = 0 и далее после прибавления 11 будет 11). Прибавляется единица и 11, а не 10, так как происходит нахождение системы счисления числа, а система на единицу больше, чем самая большая цифра числа. После обновления системы счисления (если в итоге текущая цифра больше текущей системы) символ записывается в строку «number».

Когда все символы лексемы обработаны происходит перевод числа в десятичную систему счисления с помощью функции «transform». В эту функцию передается строка «number», полученная система счисления и счетчик «k», который показывает длину записанной лексемы. В цикле «for» происходит обработка строки. Если текущий символ — это цифра, то происходит вычитание из символа `0`, а затем полученный символ умножается на систему счисления в определенной степени (зависит от индекса текущего символа). Если текущий символ — это буква, то сначала она становится строчной, а затем происходит вычитание из символа 'a' (получение ASCII-кода) и прибавление 10, что позволяет получить число, соответствующее букве. Далее происходит то же самое, что и с цифрой. Если текущий символ не является ни буквой, ни цифрой, то цикл завершается. Переведенное число возвращается.

После получения числа происходит проверка на ведущие нули (в цикле получается индекс, с которого нужно начать печать, чтобы не печатать ведущие нули). Далее, если введенное число – это просто «0», то он выводится отдельно. Иначе (результат уже будет ненулевой) в файл печатается число, его система счисления и переведенное в десятичную систему счисления число. После этого строка «number» очищается и память выделяется заново, чтобы считать новое число. После обработки всех чисел строка с числами очищается, как и строка «number».

После завершения работы функции на экран выводится либо сообщение об успешном завершении, либо сообщение об ошибке, связанной с выделением памяти. Затем открытые файлы закрываются, и программа завершается.

Задание №9. Псевдослучайное заполнение массивов

Условие:

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: для отслеживания ошибок используется тип данных перечисление, в котором определены возможные ошибки, которые могут возникнуть в процессе обработки. В программе объявлено перечисление с именем «errors», где определены значения: «success» (успешное завершение), «wrong_amount» (неверное количество аргументов), «wrong_input» (неверные аргументы), «memory_error» (ошибка выделения памяти).

Первоначально производится проверка на количество введенных аргументов, далее проверка введенных чисел с помощью функции «check_digit», которая проверяет, чтобы в записи переданного ей числа не было лишних символов кроме символов цифр. Если проверки завершились успешно, то программа продолжает работу, не завершаясь. Выделяется память для строки «numbers», куда будут записаны числа, полученные для первого пункта задания. Если левая введенная граница больше, чем правая, то их значения меняются между собой для корректности диапазона значений.

Далее происходит вызов функции «fill_random», куда передается строка, границы, а также количество чисел, записанное в переменной «length». В данной функции с помощью цикла «for» строка заполняется случайными значениями из переданного диапазона. Очередное значение получается благодаря функции «rand ()», которая генерирует случайные числа. Так как задан диапазон значений, то для ограничения получаемых значений «сверху» получается остаток от деления «rand ()» на разность границ с прибавленной единицей [6]. К полученному значению прибавляется левая граница диапазона.

После заполнения массива полученная строка выводится. Далее происходит изменение мест максимального и минимального элемента массива. Это происходит с помощью функции «swap_max_min», куда передается массив и его размер. В данной функции перед циклом «for» инициализированы переменные, отвечающие за максимальный элемент и его индекс, минимальный элемент и его индекс. Изначально индексы равны нулю, а максимальный элемент – -100000, минимальный – 100000 (благодаря этому будет возможно запомнить минимальный и максимальный элемент). В цикле идет проверка всех элементов массива и обновление максимального и минимального элементов, их индексов. После прохода по массиву происходит изменение элементов местами. Для этого во временную переменную сохраняется максимальное число, затем на место максимума записывается минимум, а на место минимума – сохраненное значение максимума. Затем измененный массив выводится на экран, а массив очищается.

Для получения размеров массивов A и B для второго пункта задания используется функция «get_random_num», куда передаются границы диапазона, заданного по условию. Данная функция возвращает псевдослучайное число так же, как это происходит в функции «fill_random» при получении числа. После получения размера выделяется память для двух массивов, а также память для массива C (его размер равен размеру массива A, так как этот массив заполняется числами из A). После успешного выделения памяти происходит заполнение массива A и B псевдослучайными числами с помощью функции «fill_random». Далее вызывается функция «fill_arr_c», которая заполняет массив C. В функцию передаются три массива, а также длина A и B.

В данной функции в цикле «for» происходит получение очередного элемента массива. Элементом будет элемент массива A с прибавленным к нему ближайшим элементом из массива B. Поиск ближайшего элемента происходит с помощью функции «find_elem», куда передается текущий элемент A, массив B и длина этого массива.

В функции «find_elem» осуществляется проход по массиву B с помощью цикла «for». Каждый раз происходит проверка очередного числа по модулю и разницы (изначально она равна 10000, чтобы была возможность обновить его в цикле). Если разность числа и элемента (переданное значение из массива A) меньше, чем текущая разница, то

разница обновляется, а число запоминается. Если разница равна нулю, то цикл завершается сразу. Найденное число возвращается.

После заполнения массива C все три массива выводятся на экран, а затем очищаются, программа завершается.

Задание №10. Обработка чисел в заданной системе счисления

Условие: пользователь вводит в консоль основание системы счисления (в диапазоне [2..36]) и затем строковые представления целых чисел в системе счисления с введённым основанием (цифры со значением больше 9 должны вводиться как прописные буквы латинского алфавита). 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: для отслеживания ошибок используется тип данных перечисление, в котором определены возможные ошибки, которые могут возникнуть в процессе обработки. В программе объявлено перечисление с именем «errors», где определены значения: «success» (успешное завершение), «wrong_input» (неверные аргументы), «memory_error» (ошибка выделения памяти).

Первоначально проверяется введенная пользователем система счисления. Если введенные данные не являются числом, а также не входят в диапазон от 2 до 36, то программа возвращает ошибку. Далее осуществляется прием чисел от пользователя. Это происходит с помощью цикла «do-while» (он работает, пока на вход не поступит строка «Stop»). В цикле принимаются строки от пользователя. Принятые числа проверяются в функции «check_elem» на то, чтобы каждая цифра числа была меньше введенной ранее системы счисления. Если это не так, то число пропускается. Иначе в переменную «length» записывается длина полученной строки, а само число переводится в десятичную систему счисления с помощью функции «to_10».

В данной функции сначала проверяется то, что переданная строка не является словом «Stop». Далее в цикле пропускаются ведущие нули. Затем в другом цикле происходит перевод числа в десятичную систему. Процесс происходит аналогично функции «transform», описанной в 8 задании.

После перевода числа происходит проверка на то, является ли оно максимальным среди введенных чисел. Если это так, то число запоминается, как и размер строки.

После обработки введенных чисел выделяется память для строки, где будет хранится записанный максимум. Сначала он выводится в десятичной системе счисления, а потом в оригинальной (в которой он был передан изначально). Для перевода числа обратно в его систему счисления используется функция «to_18_36_base», куда передается число, место для записи и основание системы счисления.

В данной функции используется цикл «while», который работает пока переданное число не равняется нулю. Если остаток от деления числа на основание системы счисления больше или равен десяти, то нужно обработать число как букву. Получается, в ответ записывается буква: берется остаток от деления числа на систему счисления, далее от полученного числа также берется остаток от деления на 10 и к нему прибавляется `a`. Таким образом, мы получим букву. Иначе просто берется остаток от деления числа на систему счисления и к этому прибавляется `0`. Затем число делится на 10, а счетчик «i», который отвечает за позицию в итоговой строке, увеличивается на единицу. После завершения цикла этот счетчик возвращается.

После обработки числа происходит его посимвольная печать в цикле. Печать происходит с конца строки, индекс элемента, с которого начинается печать, возвращается из функции перевода числа. Затем строка с числом очищается.

Далее происходит перевод числа в систему счисления с основанием 9. Это происходит в функции «to_9_base». Логика работы данной функции похожа на другие функции перевода чисел. Отличие состоит в том, что остаток от деления (и собственно деление числа) находится при делении числа на 9. После получения числа оно переворачивается в следующем цикле. В этом цикле получается цифра числа, которая прибавляется к результату. Результат каждый раз умножается на 10, а число делится на 10. Из функции возвращается результат, который затем выводится на печать.

Далее происходит получение числа в системах счисления с основаниями 18, 27 и 36. Для этого выделяется память для хранения соответствующих значений, а сами значения получаются в результате работы функции «to_18_36_base», работа которой была описана ранее.

После получения каждого результата строка с соответствующим результатом очищается.

Список заданий №2

Задание №1. Обработка строки

Условие: 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 нечётной 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, четвёртой, 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: для отслеживания ошибок используется тип данных перечисление, в котором определены возможные ошибки, которые могут возникнуть в процессе обработки. В программе объявлено перечисление с именем «errors», где определены значения: «success» (успешное

завершение), «wrong_input» (неверные аргументы), «wrong_flag» (неправильный флаг) и «memory_error» (ошибка выделения памяти).

Первоначально проверяется количество введенных аргументов. Если всё введено правильно, то происходит получение длины введенной строки. Это происходит с помощью функции «get_length».

В этой функции используется цикл «while». Пока очередной символ строки не является символом конца строки, счетчик увеличивается на единицу. После завершения цикла значение счетчика возвращается.

Далее происходит обработка флага.

Флаг «-l». Сначала проверяется количество введенных аргументов (так как для всех флагов кроме «-c» требуется два аргумента). Далее печатается длина строки, которая была получена ранее.

Флаг «-r». Также, как и в предыдущем флаге, происходит проверка количества аргументов. Далее выделяется память для хранения перевернутой строки, затем вызывается функция «reverse». В данной функции в новую строку записываются символы старой строки, но символы старой строки берутся, начиная с конца. Это осуществляется с помощью того, что индекс элемента старой строки задан как разность длины строки, текущего индекса новой строки и единицы. После завершения функции полученная строка выводится на экран.

Флаг «-u». Происходит проверка количества аргументов. Далее выделяется память для новой строки и вызывается функция «get_up». В данной функции происходит посимвольная обработка строки с помощью цикла «for». Если текущий символ строки — это буква, и текущий индекс кратен 2, то из символа вычитается 32 (меняем ASCII-код символа). После завершения функции новая строка выводится на экран.

Флаг «-n». Происходит проверка количества аргументов. Далее выделяется память для новой строки и вызывается функция «flag_n». В данной функции изначально выделяется память для строк, где будут храниться цифры, буквы и остальные символы. Дальше в цикле «for» происходит посимвольная обработка строки. Символы записываются в соответствующие им строки. После обработки строки с помощью циклов «for» происходит запись в итоговую строку сначала символов из строки с цифрами, затем с буквами, а далее с оставшимися символами. После этого использованные строки очищаются, а итоговая строка выводится на экран.

Флаг «-c». Происходит проверка введенного «seed». Если значение меньше или равно нулю, то возвращается ошибка. Иначе выделяется память для массива строк. Далее происходит запись строк из входной строки в массив. После этого вызывается функция «flag_c». В данной функции используется «srand» для получения псевдослучайных чисел. Затем в цикле получается псевдослучайный индекс (с помощью «rand()», принцип получения был описан ранее), строки меняются местами в зависимости от полученного индекса. Далее выделяется память для строки с результатом, с помощью двух циклов «for» происходит посимвольная запись строк в итоговую строку. Строка с результатом возвращается из функции и выводится на печать.

Задание №2. Среднее геометрическое и возведение в степень

Условие:

1. Реализуйте функцию с переменным числом аргументов, вычисляющую среднее геометрическое переданных ей чисел вещественного типа. Количество (значение типа int) переданных вещественных чисел задаётся 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: для получения среднего геометрического чисел вызывается функция «get_geom». В данную функцию подается количество чисел, а также сами числа. В функции осуществляется проверка введенного количества: если оно меньше или равно нуля, то функция возвращает ошибку. При успешной проверке создается переменная типа «va_list», которая является списком аргументов. Далее происходит установка указателя на первый элемент списка с помощью «va_start». В цикле происходит обработка все переданных чисел. Очередное число получается с помощью «va_arg», куда передается список аргументов, а также их тип данных. Если число равняется 0, то значит среди чисел содержится неподходящее, происходит завершение с помощью «va_end» и возврат ошибки. Иначе «va_end» вызывается после завершения цикла, а посчитанный результат возвращается. Само среднее геометрическое находится как произведение всех поданных чисел в степени 1, деленный на количество переданных чисел. Полученный результат выводится на экран.

Возведение в степень осуществляется в функции «quick_pow», куда передается число и степень. Если число равняется 0, то оно сразу возвращается. Если степень равна нулю, то возвращается 1. Иначе происходит возведение в степень. Используется бинарное возведение в степень [7]: если текущая степень четная, то вызывается функция «quick_pow», куда уже передается число и степень, поделенная на два, а результат сохраняется во временную переменную «num, который затем возвращается, умноженный сам на себя; если степень нечетная, то происходит то же самое, только результат еще умножается на изначальное число, так как в случае нечетной степени происходит переход к предыдущей степени, то есть происходит домножение, чтобы не потерять степень; если степень отрицательное, то возвращается 1 деленная на результат. Полученный результат также выводится на экран.

Задание №3. Поиск подстроки в строке

Условие: 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: обработка файлов осуществляется с помощью функции «get_file». В данную функцию передается подстрока и файлы, после которых идет строка «end», которая означает конец перечисления. В функции также используются функции из библиотеки «stdarg.h», а именно «va_list», «va_start», «va_arg», «va_end», принцип работы которых был описан в предыдущем задании. В цикле «while», пока не был достигнут конец перечисления, происходит получение очередного имени файла. Если файл открыт, то вызывается функция «find_sub», куда передается файл, подстрока и имя файла, иначе возвращается ошибка.

В функции «find_sub» происходит построчная обработка переданного файла. Пока не был достигнут конец файла происходит обработка очередной строки. Длина строки записывается в переменную «line_length», а далее происходит посимвольная обработка строки с помощью двух циклов «for», один из которых идет до достижения длины строки, а второй – до длины подстроки. Во внутреннем цикле происходит проверка текущего символа и символа подстроки. Обработка происходит, пока не закончится подстрока. То есть текущий символ сравнивается с символом подстроки. Если найдено несовпадение, то обработка начинается заново, с начала подстроки. Если прерывания не произошло и цикл дошел до конца подстроки, то в функцию для вывода информации на экран «print» передается имя файла, номер строки (этот счетчик увеличивается на единицу с получением новой строки) и позиция (индекс символа начала подстроки в строке). После завершения обработки строка, где хранилась строка файла, очищается.

Функция «print» выводит на экран информацию о найденном вхождении согласно условию задания.

После обработки всех файлов программа завершается.

Задание №4. Определение выпуклости многоугольника и значения многочлена

Условие:

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: проверка многоугольника на выпуклость происходит в функции «check_convex». В эту функцию передается количество вершин многоугольника, а также координаты точек, которые являются массивами координат типа «double». Для обработки переменного числа аргументов

также используются функции «va_list», «va_start», «va_arg», «va_end». Изначально в переменные «first_dot» и «second_dot» записываются координаты одной точки (в первую координаты по «x», во вторую – по «y»). Дальше в цикле «for» происходит обработка всех сторон, проверка на выпуклость осуществляется с помощью векторного произведения [8]. Сначала получается новая точка из списка. Далее инициализируются векторы с помощью разности координат. Затем вычисляется векторное произведение. Это нужно для того, чтобы проверить выпуклость. Если произведение больше нуля, то флаг увеличивается на единицу, иначе – уменьшается. Если после обработки всех точек флаг не равняется разности количества сторон и 2 или разности 2 и количества сторон (проверка на совпадение знаков всех полученных произведений), то многоугольник не выпуклый, иначе – выпуклый.

Получение значения полинома происходит в функции «polynom», куда передается точка, степень и коэффициенты. Так как функция также принимает переменное число аргументов, то в ней используются функции «va_list», «va_start», «va_arg», «va_end». В цикле «for», пока «i» не будет равен переданной степени, происходит сложение результата, умноженного на точку, и текущего коэффициента (то есть просто подсчет результата полинома с подставленной точкой). После завершения функции результат возвращается.

Все полученные результаты выводятся на экран, и программа завершается.

Задание №5. Реализация fprintf и sprintf

Условие: 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 в поток вывода целого числа типа int, записанного римскими цифрами;
- %Zr – печать в поток вывода цекендорфова представления целого числа типа unsigned int (коэффициенты 0 и 1 при числах Фибоначчи должны быть записаны от младшего к старшему слева направо с дополнительной единицей в конце записи, репрезентирующей окончание записи);

- %Cv печать целого числа типа int в системе счисления с заданным основанием (при обработке флага первым параметром функции, “снимаемым” со стека, является целое число типа int, вторым - основание целевой системы счисления в диапазоне [2..36] (при основании системы счисления, не входящем в диапазон, значение основания системы счисления устанавливается равным 10)); символы букв в результирующем строковом представлении целого числа должны быть записаны в нижнем регистре;
- %CV – аналогично флагу %Cv, при этом символы букв во входном строковом представлении целого числа должны быть записаны в верхнем регистре;
- %to – печать в поток вывода результата перевода целого числа, записанного в строковом представлении в системе счисления с заданным основанием в систему счисления с основанием 10 (при обработке флага первым параметром функции, “снимаемым” со стека, является строка, описываемая значением типа char *, вторым - основание исходной системы счисления в диапазоне [2..36] (при основании системы счисления, не входящем в диапазон, значение основания системы счисления устанавливается равным 10)); символы букв во входном строковом представлении целого числа должны быть записаны в нижнем регистре;
- %TO - аналогично флагу %to, при этом символы букв во входном строковом представлении целого числа должны быть записаны в верхнем регистре;
- %mi – печать дампа памяти (байты значения, записанные в системе счисления с основанием 2, в порядке нахождения в памяти “слева направо”; строковые представления байтов должны сепарироваться одним символом пробела) значения знакового целого 4-байтного числа;
- %mu – печать дампа памяти (байты значения, записанные в системе счисления с основанием 2, в порядке нахождения в памяти “слева направо”; строковые представления байтов должны сепарироваться одним символом пробела) значения беззнакового целого 4-байтного числа;
- %md – печать дампа памяти (байты значения, записанные в системе счисления с основанием 2, в порядке нахождения в памяти “слева направо”; строковые представления байтов должны сепарироваться одним символом пробела), значения вещественной переменной типа double;
- %mf – печать дампа памяти (байты значения, записанные в системе счисления с основанием 2, в порядке нахождения в памяти “слева направо”; строковые представления байтов должны сепарироваться одним символом пробела), значения вещественной переменной типа float.

Продемонстрируйте работу реализованных функций.

Решение: функция «overfprintf» принимает на вход файл (то есть поток), строку с форматами, а дальше какие-то значения. Так как число значений неизвестно, то в функции используются «va_list», «va_start», «va_arg», «va_end». С помощью цикла «for» происходит обработка строки с форматами. Если очередной символ – это «%», то значит, что это какой-то спецификатор типа. Он проверяется на то, является ли это какой-то встроенный спецификатор или нет. Если это не встроенный спецификатор, то вызывается функция «change_flags», куда передается поток, аргументы, и флаг. В этой функции происходит обработка флагов, заданных по условию задания. Иначе прибавляется счетчик, который отвечает за количество считанных элементов, а текущий символ сохраняется в строку «result». Также, если флаг не является стандартным, а счетчик «i» не равняется нулю (то есть уже были обработаны какие-то флаги), то необходимо вывести в поток уже сохраненную информацию. Это происходит с помощью функции «vfprintf», в которую передается поток, строка с записями и аргументы. Также, если флаг равняется «f», то нужно учитывать, что тип данных должен быть «double». После завершения обработки строки форматов, если какие-то записи остались неотправленными в поток (об этом говорит счетчик «j», отвечающий за индекс в строке с записями, который должен равняться нулю после обработки), то записи также отправляются в поток с помощью функции «vfprintf». Затем строка очищается, и функция завершает работу.

Функция «oversprintf» работает аналогичным образом, но на вход ей поступает буфер, а не файл. Отличие состоит в том, что в начале работы функции создается временный файл, куда будут записано то, что оказывается в строке «result». После завершения обработки строки форматов происходит разворот потока (чтобы можно было записать данные из него в буфер, начиная с начала), происходит запись всех символов в буфер, файл закрывается, а строка «result» очищается. Функция завершает работу.

Функция «change_flags» производит вызов функций в зависимости от того, какой флаг из нестандартных был встречен в строке.

Флаг «%Ro». Перед работой функции происходит выделение памяти для строки с результатом, а затем вызывается функция «flag_ro», куда передаются аргументы и строка для записи результата. В функции определены римские числа в одной строке и их значения в другой. Пока число больше нуля происходит его обработка. Если число больше или равно

текущему значению в строке значений и если текущий индекс строки значений четный, то в результат записывается соответствующая римская цифра, а число уменьшается на соответствующий результат. Если индекс нечетный (то есть римское число состоит их двух знаков), то знаки записываются по очереди, а число также уменьшается на соответствующее значение. Иначе проверяется следующее число из строки значений.

Флаг «%Zr». Обработка осуществляется в функции «flag_zr». Если число равно нулю, то возвращается «01». Иначе сначала получается длина итоговой строки с помощью функции «get_fib». В этой функции происходит последовательный поиск чисел Фибоначчи, а также их подсчет. Их количество и возвращается из функции. После получения длины выделяется память для строки, где будет храниться результат. Далее в цикле результат заполняется нулями и финальными единицами. Далее, пока число не равняется нулю, происходит изменение нулей на единицы на тех индексах, которые равняются «count» (так как count по сути является индексом, где было получено последнее число Фибоначчи). Из числа каждую итерацию вычитается последнее полученное число Фибоначчи.

Флаги «%Cv» и «%CV». Обработка происходит с помощью функции «flag_cv», куда передается число, основание системы счисления, строка для записи результата и флаг. Если переданное число равняется нулю, то возвращается ноль. Иначе происходит перевод числа в необходимую систему счисления аналогично тому, как это происходило в предыдущих заданиях. Отличие состоит в том, что проход осуществляется с конца строки, чтобы не переворачивать строку после обработки, а также в том, что если передан флаг 0 (это %Cv), то в записи будут строчные буквы, иначе – заглавные.

Флаги «%to» и «%TO». Обработка осуществляется с помощью функции «flag_to», куда передается число в строковом представлении, основание системы счисления и флаг. Данная функция переводит число в десятичную систему счисления так, как это было в предыдущих заданиях. Отличие такое же, как в функции «flag_cv» - изменение в зависимости от переданного флага.

Флаги «%mi», «%mu», «%md», «%mf». Обработка осуществляется в функции «flag_m», куда передается число соответствующего типа данных, размер (в зависимости от используемого флага: «signed int», «unsigned int», «double» или «float»), строка для записи результата. В функции сначала создается указатель «byte», который указывает на переданное число, далее выделяется память для результата (результат будет содержать биты,

поэтому размер умножается на 8, так как в байте содержится 8 бит). Далее начинаются циклы «for», в которых внешний цикл работает с байтами введенного числа, а внутренний – с битами, который обрабатывает, начиная со старшего. В строку с результатом записывается значение бита, которое находится с помощью сдвига бита числа и получения младшего бита благодаря «&1».

Также в программе присутствует функция «print», которая получает какое-то состояние и в зависимости от него выводит сообщение о соответствующей ошибке.

Задание №6. Реализация fscanf и sscanf

Условие: 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 (коэффициенты 0 и 1 при числах Фибоначчи должны быть записаны от младшего к старшему слева направо с дополнительной единицей в конце записи, репрезентирующей окончание записи);
- %Cv считывание из потока ввода целого числа типа int, записанного в системе счисления с заданным основанием (при обработке флага первым параметром функции, “снимаемым” со стека, является адрес места в памяти типа int *, куда необходимо записать считанное значение, вторым - основание системы счисления (в которой записано число, находящееся в потоке ввода) в диапазоне [2..36] (при основании системы счисления, не входящем в диапазон, значение основания системы счисления устанавливается равным 10)); символы букв во входном строковом представлении целого числа должны быть записаны в нижнем регистре;
- %CV – аналогично флагу %Cv, при этом символы букв во входном строковом представлении целого числа должны быть записаны в верхнем регистре;

Валидация данных, находящихся во входном потоке, не требуется. Продемонстрируйте работу реализованных функций.

Решение: функция «overfscanf» получает на вход файл, строку с форматами и значения (аргументы). Логика обработки соответствует логике

работы функции «overfprintf» из предыдущего задания. Отличие состоит в использовании «vfscanf» вместо «vfprintf», так как в этом задании происходит считывание, а не запись. Аналогично с функцией «oversscanf», которая работает аналогично функции «oversprintf».

Обработка флагов происходит в функции «change_flags». В ней сначала получается строка из файла, поданного функции, с помощью функции «get_file_string». В данной функции происходит посимвольный разбор строки файла (пока не был встречен символ конца файла). Символы записываются в переданную функции строку.

Флаг «%Ro». Обработка происходит с помощью функции «flag_ro», которая принимает на вход число в строковом представлении. В цикле «for» происходит посимвольная обработка. Каждый полученный символ передается в функцию «roman_numbers», которая возвращает значение, соответствующее данной римской цифре. Если текущий результат больше предыдущего, то к итоговому результату прибавляется разность текущего числа и предыдущего, умноженного на два. Иначе к результату прибавляется просто полученное число.

Флаг «%Zr». Обработка происходит с помощью функции «flag_zr», куда передается строковая запись числа. С помощью цикла «while» происходит посимвольная обработка числа. Текущее число Фибоначчи получается, как сумма двух предыдущих. Если текущий символ строки является единицей, то это число прибавляется к результату.

Флаги «%Cv» и «%CV». Обработка осуществляется с помощью функции «flag_cv». В функции происходит перевод числа в десятичную систему счисления аналогично функции «flag_to» из предыдущего задания.

Также в программе присутствует функция «print», аналогичная такой же функции из предыдущего задания.

Задание №7. Вычисление корня уравнения методом дихотомии

Условие: 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: для проверки введенных границ вычисления используется функция «check_border», которая проверяет строковую запись чисел на

отсутствие лишних символов. Аналогично происходит для эпсилона в функции «check_eps».

Для вычисления корня уравнения используется функция «dichotomy», куда передаются границы, эпсилон и уравнение. Согласно методу [9], изначально сохраняется середина отрезка (в переменной «x»), а далее происходит вычисление результата в цикле «while». Цикл длится пока очередное значение не приближено к эпсилону максимально. В цикле происходит вычисление очередного результата в граничных точках и в точке середины отрезка. Если очередное значение больше нуля, то оно становится левой границей, иначе – правой. Значение середины отрезка, соответственно, также обновляется.

Также в программе содержатся функции, в которых записаны примеры уравнений. Они передаются в функцию «dichotomy» для нахождения их результата.

Задание №8. Вычисление суммы чисел

Условие: 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: суммирование происходит в функции «total_sum», куда передается основание системы счисления, количество чисел и числа. Так как количество аргументов переменно, то в функции используются функции из библиотеки «stdarg.h». В цикле «for» происходит обработка чисел. Сначала число проверяется с помощью функции «check_number» на то, чтобы его цифры не были больше или равны введенной системы счисления. После успешной проверки числа вызывается функция «sum_in_st», где происходит прибавления очередного числа к результату.

В данной функции сначала получается длина переданного результата и числа. С помощью функции «remove_zeros» из числа удаляются ведущие нули путем перезаписи результата со сдвигом. После обновления длин происходит вычисление максимальной длины (для выделения памяти для результата). В цикле «for» происходит поочередное получение цифры текущего результата и числа. Далее происходит суммирования их и остатка (остаток является результатом деления суммы и системы счисления; сохранения остатка нужно, так как при суммировании цифр у нас может получиться результат больше 9, а в результат сумма записывается по одной цифре). То есть в результат записывается самая правая цифра, а самая левая сохраняется в остаток. После получения результата он записывается в итоговую строку и из него удаляются ведущие нули, если они остались после суммирования.

Результат после возвращения выводится на экран.

Задание №9. Проверка дроби на периодичность

Условие: 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: обработка происходит в функции «fraction», куда передается основание системы счисления и дроби. В цикле «while» происходит обработка каждой дроби. Если дробь больше нуля и меньше единицы, то происходит проверка её на периодичность в функции «is_period», куда передается дробь и основание системы счисления.

С помощью функции «modf» из дроби выделяется целая и дробная часть. Пока дробная часть числа больше эпсилона дробь умножается на 10 и из неё также берется дробная и целая часть. Таким образом получается числитель дроби. Далее происходит получение знаменателя в функции «get_denumer», куда передается найденная целая часть дроби.

В данной функции изначально происходит умножение знаменателя на 10 столько раз, чему равна длина переданного числа (это нужно, чтобы поместить все цифры). Далее, пока число кратно двум и знаменатель кратен двум, происходит деление числа и знаменателя на 2 (то есть сокращение). Затем число и знаменатель делятся в цикле «for» на числа, начиная с 3 и до корня из числа. Если число в итоге больше 2 и если знаменатель кратен

числу, то знаменатель также делится на число. Иначе знаменатель возвращается.

После получения знаменателя происходит проверка, кратна ли система счисления двум. Если кратно, то также происходит деление знаменателя на 2, пока знаменатель делится. Иначе, если знаменатель кратен, а основание системы счисления нет, то возвращается неуспех. Затем в цикле «for» происходит деление знаменателя на числа от 3 до корня из знаменателя. Но если система счисления не кратна числу, которому кратен знаменатель, то возвращается «fail». То есть происходит проверка на то, что система счисления и знаменатель имеют общие делители. Если это так, то дробь не периодическая, иначе – периодическая. Соответствующий результат выводится с помощью функции «print», которая вызывается после обработки дроби, куда передается результат и дробь.

Задание №10. Нахождение переразложения многочлена

Условие: 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 $g_0, g_1, \dots, g_n$ такие что:

$$f(x) = f_0 + f_1 x + f_1 x + f_2 x^2 + \dots + f_n x^n = \\ = g_0 + g_1(x-a) + g_2(x-a)^2 + \dots + g_n(x-a)^n$$

Продемонстрируйте работу функции.

Решение: обработка происходит в функции «polynom», куда передается эпсилон, точка «a», место для записи результата, степень многочлена и коэффициенты. В функции происходит выделение памяти для результата и для хранения коэффициентов. Коэффициенты записываются в массив, а далее в цикле происходит нахождение суммы (так как в итоговом многочлене будет свободный член) одночленов путем умножения очередного коэффициента на точку «a» в соответствующей степени. Далее происходит выделение памяти для массива, где будут храниться производные. Нахождение полинома осуществляется путем нахождения

производных многочленов, поделенных на факториал [10]. В цикле происходит нахождение производной с помощью умножения коэффициента на пониженную степень. Соответственно, изменяется и сумма. После нахождения значения происходит деление её на соответствующий факториал. Факториал находится с помощью функции «fact», которая аналогична функциям нахождения факториала из предыдущих заданий. Полученное значение сохраняется. После обработки полученный многочлен выводится.

Список заданий №3

Задание №1. Перевод числа в заданную систему счисления

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

Решение: первоначально происходит получения основания системы счисления и числа от пользователя. Если входные данные введены верно, то вызывается функция «to_base», куда передается число, система счисления (она получается сс помощью сдвига бита 1 на поданную степень «r»: например, если степень равна 3, то в результате получится 8 (0001 << 3 = 1000)), место для записи результата и полученная степень.

В данной функции изначально происходит проверка поданных аргументов. Также если число равняется нулю, то сразу же возвращается 0 как результат. Иначе, пока число, записанное во временную переменную, больше нуля, то происходит получение его длины путем прибавления к «length» единицы и делением числа (с помощью сдвига вправо) для получения всех разрядов числа. Суммирование происходит с помощью функции «sum».

В данной функции, пока второе слагаемое не равняется нулю, происходит сохранения результата побитового И между двумя слагаемыми (для получения количества разрядов), далее происходит использование исключающего ИЛИ (происходит сложение без учета переноса), а затем второе слагаемое сдвигается (для получения следующего разряда). В итоге из функции возвращается измененное первое слагаемое.

После получения длины и выделения памяти для результата в цикле «for» происходит обработка числа справа налево. Сначала получается текущий бит числа, то есть значение текущего разряда, путем использования побитового И между числом и основанием системы счисления (оно понижается на единицу с помощью функции «sub»).

В функции «sub» происходит вычитание. Пока вычитаемое не равняется нулю происходит сохранение результата побитового И между уменьшаемым (которое было инвертировано, то есть также получаются разряды). Далее с помощью исключающего ИЛИ происходит вычитание, после этого происходит получение нового разряда с помощью сдвига. Измененное уменьшаемое возвращается.

Если полученный бит больше 10, то происходит запись соответствующей буквы в результат, иначе происходит запись цифры. Затем число сдвигается для получения следующего разряда.

Переведенное число выводится на экран после обработки.

Задание №2. Вычисление нормы вектора

Условие: 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 (если таковых векторов несколько, необходимо вернуть их все).

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: первоначально задается матрица как двумерный массив. Для хранения векторов используется структура «Vector», элементами которой являются массив координат вектора и его размер. Происходит создание векторов, а также выделение памяти для результатов вычислений. Далее вызывается функция «find_norm», куда передается количество векторов, адреса результатов, счетчиков, матрица, функции для вычисления норм и сами векторы.

В функции происходит вычисление разных норм. В цикле, который длится, пока не были обработаны все векторы, происходит вычисление каждой нормы, а также проверка на то, является ли она максимальной среди норм определенного вида. Если норма больше текущего максимума, то он удаляется и норма становится новым максимумом, если равна, то вектор записывается к другим векторам, у которых максимальная норма.

Получение нормы вектора происходит в функции «get_norm», куда передается вектор. Здесь происходит вычисление Евклидовой нормы вектора. В цикле происходит обработка всех координат вектора, которые возводятся в квадрат и прибавляются в итоговый результат. В итоге возвращается корень из результата.

Получение P-нормы вектора происходит в функции «get_p_norm», куда передается вектор и значение «p». Вычисление происходит также путем обработки всех координат в цикле. Очередная координата возводится в степень «p» и прибавляется к результату. Из функции возвращается результат в степени $\frac{1}{p}$.

Получение матричной нормы происходит в функции «get_matrix_norm», куда передается вектор, матрица и размер вектора. В двух циклах происходит обработка координат. К результату прибавляется текущая координата матрицы, умноженная на соответствующие координаты вектора. В итоге из функции возвращается корень из полученного результата.

После обработки всех векторов и получения максимальных норм происходит печать максимумов.

Задание №3. Реализация структуры Employee

Условие: 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: для хранения информации о сотруднике определена структура «Employee», элементами которой является «id» сотрудника, его имя, фамилия и зарплата. Происходит проверка введенного флага с помощью функции «check_flag» на то, чтобы флаг равнялся одному из допустимых флагов. Перед вызовом функции для обработки входных данных происходит инициализация массива для хранения информации о сотрудниках – «employees». Обработка данных происходит в функции «reading», куда передается входной файл, массив сотрудников и их количество, которое изначально равняется нулю.

В данной функции первоначально выделяется память для массива «employees». В последствии, если число сотрудников превысит 10, то размер будет динамически увеличен. В цикле, пока не достигнут конец входного файла, происходит построчная обработка файла. Каждая строка проверяется на наличие всей необходимой информации о сотруднике. Если количество считанных из строки элементов равняется 4, то происходит проверка имени и фамилии на правильность с помощью функции «check_name». В этой функции происходит посимвольная проверка переданной строки на то, что в ней должны содержаться только символы букв. Если все данные введены правильно, то очередной сотрудник записывается в массив, а количество работников увеличивается.

После считывания сотрудников из файла происходит вызов функции «qsort», которая сортирует массив сотрудников. Если передан флаг «a», то в качестве компаратора используется функция «compare_flag_a», иначе – функция «compare_flag_d».

Эти функции сравнивают сначала сравнивают зарплату сотрудников, далее их фамилии, имена и, наконец, «id». Соответственно, функции возвращают соответствующие числовые значения.

После сортировки сотрудников происходит печать полученного массива.

Задание №4. Реализация структуры String, Mail

Условие:

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. Продемонстрируйте работу реализованного функционала.
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: пункт 1. В программе определена структура «String», элементами которой являются строка и её длина.

Для создания строки используется функция «create», куда передается объект структуры и строка, которую необходимо записать. В функции происходит присваивание объекту новой длины с помощью получения функцией «strlen» длины переданной строки. Далее происходит выделение памяти для хранения строки объектом и копирование строки.

Удаление объекта происходит в функции «delete», где очищается память, выделенная под хранение строки.

Сравнение строк происходит в функции «compare_strings», куда передаются два объекта структуры. Сначала происходит сравнение длин, а затем сравнение записанных строк с помощью функции «strcmp». Соответствующий результат возвращается из функции.

Проверка на эквивалентность строк происходит в функции «equality», где строки сравниваются с помощью «strcmp», а соответствующий результат возвращается.

Копирование содержимого экземпляра типа «String» в другой существующий экземпляр осуществляется с помощью функции «copy_to_exist», куда передаются два объекта. Сначала происходит обновление длины, затем удаление предыдущей сохраненной строки и копирование новой в заново выделенную память.

Копирование в новый экземпляр происходит аналогично, только не происходит удаления.

Конкатенация экземпляров происходит в функции «concate», куда передаются два объекта. Сначала происходит обновление длины объекта, к

которому будет дописан другой объект. Затем происходит увеличение памяти для хранения строки и объединение строк с помощью «strcat».

Пункт 2. В программе определена структура «String», которая необходима для хранения строк (элементы структуры — это строка и её длина), структура «Post», которая содержит число посылок, указатель на массив структур «Mail», содержащий почтовые отправления и экземпляр структуры «Address», содержащей адрес. Структура «Mail» содержит также экземпляр структуры «Address», в которой находится адрес, вес отправления, идентификатор и время создания и доставки отправления. Структура «Address» содержит такие элементы как город, улица, номер дома, строение, номер квартиры и почтовый индекс.

Интерактивный диалог с пользователем реализуется с помощью печати меню и приема от пользователя соответствующего числа, которое указывает на выбранную опцию. Печать меню, а также других сообщений осуществляется с помощью функции «print», куда передается соответствующее состояние.

При добавлении посылки происходит получение от пользователя соответствующих данных. Строковые значения проверяются в функции «get_string». В данной функции создается экземпляр структуры «String», а после его заполнения происходит проверка введенного значения на валидность. Для проверки используются функции «is_valid_index», «is_valid_id» и «is_valid_time», где в зависимости от данных происходит посимвольная проверка. После получения всех данных вызывается функция «add_package», куда передаются эти данные и адрес экземпляра структуры «Post» для записи. В данной функции происходит получение корректного вида времени с помощью функции «get_time», а затем сохранение всех данных в новом объекте, увеличение числа посылок. Вставка новой посылки к другим происходит с помощью сравнения сначала почтового индекса, а затем «id» отправления.

Листинг 1. Структуры, используемые для решения задания №4

```
typedef struct{
    char *string;
    int length;
} String;

typedef struct{
    String city;
    String street;
```

```
    int house;
    String building;
    int apartment;
    String index;
}Address;

typedef struct{
    Address address_client;
    double weight;
    String id;
    String creation_time;
    String delivery_time;
}Mail;

typedef struct{
    Address post_address;
    Mail *packages;
    int packages_number;
}Post;
```

Если число посылок не равно нулю, то возможно удаление посылки. От пользователя получается «id» посылки, которую нужно удалить и далее вызывается функция «delete_package». В этой функции сначала происходит нахождение объекта с нужным «id», а далее очищение полей с помощью вспомогательной функции «delete». После очищения посылки в системе сдвигаются, так как освободилось место, а количество посылок уменьшается, размер массива отправлений также обновляется.

Получение информации о посылке происходит с помощью полученного от пользователя «id» посылки. С помощью функции «find_mail» происходит нахождение посылки в системе. В этой функции в цикле происходит проход по существующим отправлениям и сравнение их «id». Если найдено совпадение, то соответствующая посылка сохраняется. После получения посылки информация о ней выводится.

Получение доставленных посылок происходит с помощью функции «get_delivered_packages». В данной функции осуществляется проход по посылкам и получение их времени доставки с помощью функции «get_time». Далее происходит сравнение текущего времени и полученного. Если полученное время меньше текущего, то посылка была доставлена и она сохраняется. После выхода из функции сохраненные посылки выводятся.

Также возможна печать всех посылок, которая осуществляется проходом по посылкам и выводом информации о них.

Задание №5. Реализация структуры Student

Условие: 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: в программе определена структура «Student» с соответствующими полями. Получение данных из входного файла происходит в функции «reading», куда передается файл, адрес экземпляра структуры и число студентов для обновления. Данная функция аналогична функции «reading» из третьего задания: здесь также происходит построчная

обработка файла, проверка имени и фамилии на корректность в функции «check_name» и сохранение данных.

Интерактивный диалог с пользователем осуществляется аналогично четвертому заданию.

Для поиска студента сначала происходит получение от пользователя данных, по которым будет осуществляться поиск, а далее вызывается либо функция «find_student_id», либо функция «find_student_str».

В первой функции происходит проход по сохраненным студентам и сравнение «id» очередного студента и поданного «id». Также найденный студент записывается в файл. Во второй функции также осуществляется проход по студентам, но сравниваемая информация, соответственно, другая. Использование разных функций обусловлено тем, что данные подаются разных типов: целое число и строки.

При сортировке студентов используется «qsort», куда в качестве компаратора подается либо «compare_students_id», либо «compare_students_surname», либо «compare_students_name», либо «compare_students_group» в зависимости от того, сортировку относительно чего выбрал пользователь.

Получение лучших студентов происходит с помощью функции «find_best_student». Сначала происходит нахождение среднего балла среди всех студентов с помощью функции «average_grade». В этой функции суммируются оценки всех студентов, а полученная сумма делится на их количество. После получения среднего балла происходит проход по всем студентам. Для каждого студента получается его средний балл, который сравнивается с общим средним баллом. Если балл студента больше, то он сохраняется и записывается в файл.

Функция «average_grade_student» необходима для получения среднего балла студента, чтобы использовать эту информацию при записи студента в файл после нахождения.

Функция «delete_students» удаляет информацию о студентах, очищая память.

Задание №6. Реализация хранилища информации

Условие: 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: так как информация должна храниться на базе односвязного списка односвязных списков, то в программе определена структура «List» - односвязный список остановок, элементом которого является указатель на первый узел списка, структура «Stop_node» - узел списка, элементами которого является структура типа «Info» и указатель на следующий узел, структура «Info» - информация о транспорте.

Листинг 2. Структуры, используемые для решения задания №6

```
typedef struct{
    char *time_stop;
    char *time_start;
    char *number;
    char mark;
    double coords[2];
}Info;
```

```
typedef struct Stop_node{
    Info stop;
    struct Stop_node *next;
}Stop_node;

typedef struct List{
    Stop_node *head;
}List;
```

Считывание информации из файла происходит с помощью функции «read_file», куда передается файл, список, массив для хранения координат, размер (обновляемый) и число транспорта. В данной функции обработка происходит аналогично функции «reading». Также происходит проверка введенного времени на корректность с помощью функции «check_time», проверка правильности времени отправления и остановки с помощью функции «check_dif» (так как время отправления должно быть больше времени остановки). Функция «create_stop» осуществляет сохранение информации о транспорте, а функция «insert» помещает новый транспорт в список в зависимости от его времени.

Интерактивный диалог с пользователем также осуществляется с помощью вывода меню. Получения результата функций происходит в функции «change_command», где в зависимости от выбранной пользователем функции вызываются соответствующие функции.

Получение транспорта, которое проехало больше/меньше всех маршрутов происходит с помощью функции «find_routes_number», куда передается список и адреса переменных для сохранения результатов. В этой функции происходит обработка каждого транспорта для каждой остановки. Для него считается, сколько остановок были начальными и конечными. Если их число совпало (то есть все маршруты являются законченными), то происходит проверка, является ли это число маршрутов максимумом/минимумом или нет.

Получение транспорта, которое проехало самый длинный/короткий путь происходит в функции «find_longest_way». Логика работы похожа на функцию «find_routes_number», но здесь учитывается сумма всех пройденных остановок. Соответственно, полученная сумма сравнивается с минимумом и максимумом.

Нахождение транспорта с самым длинным/коротким маршрутом происходит в функции «find_longest_route». Принцип схож с предыдущими функциями, но здесь проверяется количество промежуточных точек между

началом и концом маршрута, благодаря чему находится минимум и максимум.

Поиск транспорта с самой длинной/короткой остановкой происходит в функции «get_time». Для получения времени простоя на остановке используется функция «get_duration». В этой функции вызывается функция «convert_time», которая из строкового представления времени получает корректное, а затем происходит нахождение разницы времени остановки и отправления с помощью функции «difftime». Таким образом полученная разница сравнивается с максимумом и минимумом для получения результата.

Получение транспорта с самым большим временем простоя происходит в функции «find_longest_stop». В этой функции также происходит получение времени простоя, но если в прошлой функции время получалось для каждой остановки, то здесь находится самое большое время для всех остановок, но логика обработки такая же.

Функция «free_list» удаляет список, очищая использованную память.

Задание №7. Реализация структуры Liver

Условие: 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 последние $N/2$ введённых модификаций, то есть аналог команды Undo; N - общее количество модификаций на текущий момент времени с момента чтения файла/последней отмены введённых модификаций.

Решение: в программе определена структура «List», элементами которой является указатель на первый элемент списка, а также указатель на

экземпляр структуры «Changes», структура «Changes», которая нужна для сохранения произошедших изменений и содержит также указатель на следующий элемент списка, структура «Liver», которая содержит информацию о жителе и указатель на следующий элемент списка.

Листинг 3. Структуры, используемые для решения задания №7

```
typedef struct Liver{
    char *name;
    char *surname;
    char *patronymic;
    int day;
    int month;
    int year;
    char gender;
    double salary;
    struct Liver *next;
}Liver;

typedef struct Changes{
    int option;
    int operation;
    Liver *old_state;
    char *old_value;
    double old_salary;
    int old_date;
    char old_gen;
    struct Changes *next;
}Changes;

typedef struct{
    Liver *head;
    Changes *change_history;
}List;
```

Обработка полученного файла происходит в функции «read_file», которая аналогична таким же функциям из предыдущих заданий. Для создания жителя используется функция «create_liver», где происходит проверка переданных данных на валидность (проверка имени, фамилии и отчества происходит с помощью функции «check_name»), а далее сохранение данных. Помещение жителя в список происходит с помощью функции «add_liver_to_list», где житель «встает» на свободное место.

Поиск жителя происходит в функции «find_liver». Каждый элемент списка сравнивается с переданными данными. В случае совпадения житель запоминается и возвращается успех. Иначе, если житель не был найден, то возвращается соответствующий статус-код.

При изменении информации о жителе происходит сначала проверка, есть ли этот житель в списке (с помощью функции «find_liver»), а далее происходит получение новой информации и изменение в функции «change_liver», куда передается житель, флаг (указывает на то, что нужно поменять у жителя) и лист. Считывание новой информации происходит внутри функции. Старая информация удаляется, а на её место записывается новая. Перед изменением житель сохраняется с помощью функции «copy_liver» (эта функция полностью копирует информацию о жителе в другой объект структуры). После изменения информации вызывается функция «add_changing». Эта функция сохраняет старого жителя (с помощью того же копирования), а также сохраняет, что именно было изменено.

Удаление жителя происходит с помощью функции «remove_liver», где житель находится в списке (с помощью функции «check_identity» проверяется найден он или нет, так как в функции происходит сравнение информации текущего жителя и переданной). Когда житель найден происходит сдвиг списка и удаление жителя. Удаление происходит с помощью «free_liver», где очищается память, использованная для хранения.

Запись информации о жителях в файл происходит в функции «write_to_file», где в цикле происходит обработка всех элементов списка и печать их в переданный файл.

Функция «undo» отменяет проделанные операции. В нее передается список и количество операций, необходимых к отмене. Поле «change_history» содержит сделанные изменения. В цикле, пока не был получен конец истории изменений или нужное число отмен не было выполнено. Сама отмена происходит в функции «undo_change». Если изменение – изменение информации о жителе, то происходит копирование жителя (сохраненного ранее) в переменную, изменение его поля обратно с помощью функции «change_back», удаление измененного жителя из списка и возвращение старого. Если изменение – удаление жителя, то он также копируется в переменную и возвращение его в список. Если изменение – добавление жителя, то он удаляется из списка. Далее происходит вызов функции «free_changes», которая удаляет сохраненные изменения, так как они уже были отменены.

Печать списка происходит в функции «print_list», где в цикле выводится информация о жителях. Очистка списка производится в функции

«free_list», где очищается использованная память, а также очищается история изменений.

Задание №8. Обработка многочленов

Условие: 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: в программе определена структура «Polynom», элементами которой является ссылка на первый элемент списка с одночленами, ссылка на следующий полином и первый элемент списка мономов, структура «Monom», элементами которой является коэффициент и степень монома, а также ссылка на следующий элемент списка.

В функции «create_polynom» происходит инициализация элемента списка. Функция «add_monom» создает новый моном в переданном полиноме, учитывая поданный коэффициент и степень.

После инициализации сумматора происходит построчная обработка входного файла.

Строка проверяется на валидность с помощью функции «check_string». В данной функции проверяется правильность входной строки (все скобки должны быть закрыты, в конце строки должен быть символ «;», комментарии не должны быть вложенными). Соответственно, функция возвращает либо статус-код «success» при успешной проверке, либо «wrong_input» в противном случае.

В функции «read_string» происходит разбор строки на команду и полиномы. С помощью функции «strtok» происходит разделение строки сначала на команду и оставшуюся часть (полиномы). Полученная команда проверяется с помощью функции «command_check» (из этой функции возвращается статус-код, соответствующий команде, иначе возвращается ошибка). Далее происходит обработка полинома в функции «get_polynom».

В данной функции первоначально выделяется память для хранения коэффициентов и степеней. Обработка строки происходит справа налево. Строка посимвольно разбирается и соответствующие значения запоминаются. Если текущий символ — это операция, то сохраненные значения преобразуются в моном, который добавляется в полином. Перед добавлением степени в результат строка со степенью переворачивается с помощью функции «reverse», так как в строку может заносится символ «^». Если данный символ останется в начале строки, то «atoi» не выдаст верный результат для степени. Поэтому строка переворачивается. После добавления монома строки очищаются и проход продолжается. Полученный полином сортируется с помощью функции «sort_poly» (для того, чтобы слева направо в полиноме сначала шли старшие степени, а затем младшие). В данной функции в цикле происходит изменение порядка мономов в полиноме.

После обработки полинома он возвращается и помещается в список, происходит обработка второго (если он подан). Далее вызывается функция «change_command», где выполняется нужная операция над полиномами. В функцию подается команда, список полиномов, сумматор и количество полиномов.

В данной функции с помощью конструкции «switch-case» происходит вызов необходимых функций. Общим для всех случаев является то, что сначала проверяется, сколько подано полиномов. Если полином один, то вместо второго в функцию подается сумматор. После успешного завершения функции происходит копирование результата в сумматор, его сортировка, вывод результата. Далее сумматор сохраняется, а полиномы очищаются для записи новых.

Сложение полиномов происходит в функции «add_poly». В данной функции с помощью цикла происходит проход по полиномам. Если степень текущего монома первого многочлена больше, чем второго, то первый моном добавляется в результат. Иначе добавляется второй. Если мономы равны, то коэффициенты складываются, и соответствующий моном добавляется в результат. Цикл длится до конца одного из полиномов. Если другой полином не закончился, то его оставшиеся мономы также добавляются в результат.

Вычитание полиномов происходит в функции «sub_poly». Данная функция аналогична функции «add_poly». Отличие состоит в том, что, если степень монома вычитаемого больше, чем уменьшаемого, то в результат добавляется моном, умноженный на -1. То же самое происходит, если после окончания цикла в вычитаемом остались необработанные мономы.

Умножение полиномов происходит в функции «mult_poly». Обработка происходит в цикле, который длится до тех пор, пока не был достигнут конец первого множителя. Мономы первого полинома умножаются на каждый моном второго, результат умножения добавляется в итоговый полином. После умножения происходит упрощение полученного полинома с помощью функции «combine_monoms».

В данной функции происходит проход по полиному в цикле. Происходит сравнение текущего монома со всеми остальными в полиноме. Если находится моном с такой же степенью, то мономы складываются.

Целочисленное деление полиномов происходит в функции «div_poly». Обработка также происходит с помощью цикла, который

работает либо пока первый моном не закончился, либо пока старшая степень первого полинома больше или равна старшей степени второго. Каждый раз происходит нахождение текущего коэффициента с помощью деления коэффициента монома первого полинома на моном второго и нахождение степени с помощью вычитания. Получившийся моном добавляется в результат, а затем умножается на делитель. Далее происходит нахождение разности полученного полинома и делимого. То есть происходит деление полиномов «уголком». Полученный результат упрощается.

Нахождение остатка от деления полиномов происходит в функции «mod_poly». Функция аналогична функции «div_poly». Но результатом является итоговый вид делимого.

Вычисление многочлена в заданной точке происходит в функции «eval_poly». В цикле происходит обработка всех мономов полинома. К результату прибавляется текущий коэффициент, умноженный на поданную точку в текущей степени.

Дифференцирование многочлена происходит в функции «diff_poly». Если поданный полином – это 0, то 0 и возвращается. В цикле происходит обработка всех мономов. Текущий коэффициент умножается на степень, а затем степень уменьшается на единицу. Полученный моном добавляется в результат.

Нахождение композиции двух многочленов происходит в функции «cmps_poly». Обработка происходит в цикле, который длится, пока не закончится первый полином. Сначала происходит умножение второго полинома на себя же столько раз, чему равняется текущая степень монома первого полинома. Далее происходит умножение получившегося полинома на текущий коэффициент, а затем сложение его и результата. В конце результат упрощается.

Для очистки полинома используется функция «free_polynom».

Для печати на экран используется функция «print_poly». В данной функции с помощью цикла происходит проход по полиному и печать мономов.

Задание №9. Построение двоичного дерева поиска

Условие:

А) 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: в программе определена структура «Node», элементами которой является строка «word», где хранится слово, счетчик «count», показывающий, сколько раз слово содержится в строке, ссылка на левого потомка и ссылка на правого потомка.

Первоначально происходит заполнение строки сепараторов аргументами входной строки.

Обработка входного файла происходит в функции «read_file», куда передается файл, строка с сепараторами и адрес корня.

Данная функция посимвольно обрабатывает файл. Каждый символ проверяется в функции «is_sep» на то, является ли он сепаратором (в данной функции осуществляется проход по строке с сепараторами и сравнение символа с текущим элементом строки; если совпадение найдено, то функция вернет «success», иначе «fail»). Если символ не сепаратор, то он добавляется в строку «word». Иначе происходит добавление нового узла в дерево с помощью функции «insert». Если после обработки файла строка со словом не оказалось пустой, то слово также добавляется в дерево.

В функцию «insert» передается корень и слово. Если корень «NULL» (то есть дерево еще пустое), то корнем становится полученное слово. Иначе слово сравнивается с корнем с помощью функции «strcmp». Если слова

одинаковые, то происходит увеличение счетчика узла. Если значение, возвращенное «strcmp» меньше нуля, то слово вставляется в левую ветвь дерева (происходит вызов «insert» для левого поддерева и слова; то есть рекурсивно осуществляется проход по левому поддереву, пока не будет найдено нужное место), иначе – в правую.

После обработки файла и построения дерева вызывается функция «change_command», куда передается дерево и строка с сепараторами.

Интерактивный диалог с пользователем осуществляется по такому же принципу, который был описан ранее. То есть на экран выводится меню, а от пользователя принимаются номера, соответствующие функциям. В «change_command» происходит вызов определенных функций, пока пользователь не запросит выход из программы.

Вывод информации о том, сколько раз заданное слово встречалось в файле, происходит с помощью функции «find_word». Перед вызовом функции пользователь вводит слово, которое необходимо обработать. Это слово вместе с деревом передается в функцию. В «find_word» первоначально проверяется, пустой корень или нет. Если корень пуст, то возвращаемое значение равняется нулю. Иначе происходит сравнение слова и слова, которое хранится в узле дерева. Если слова одинаковые, то к результату прибавляется значение счетчика узла. Если результат сравнения меньше нуля, то происходит проход по левому поддереву, иначе – по правому. Из функции возвращается найденное значение счетчика.

Вывод первых «n» наиболее часто встречающихся слов в файле происходит с помощью функции «get_words». Перед вызовом функции происходит получение числа от пользователя. Если введенное число больше количества узлов в дереве (это число находится в функции «total_words», где происходит рекурсивный обход левого и правого поддерева, подсчет узлов и возврат полученного значения), то возвращается ошибка. Иначе происходит вызов соответствующей функции, куда передается дерево, двумерный массив для хранения узлов, счетчик для отслеживания количества заполненных мест в массив и число. В функции осуществляется проверка счетчика узла с текущим в массиве. Если текущее место еще не занято, то узел записывается туда. Если счетчик узла больше, чем счетчик текущего элемента массива, то происходит сдвиг элементов, а затем вставка узла. Таким образом обходится левое и правое поддеревья. Полученные слова выводятся после завершения функции.

Поиск самого длинного и самого короткого слова происходит с помощью функции «get_long_short_word», куда передается дерево и адреса

переменных, куда будут записаны найденные слова. В функции происходит сравнение длины текущего максимального и минимального слова с текущим узлом. Если текущий узел больше максимума или меньше минимума по длине слова, то слово, соответственно, сохраняется. Таким образом обходится левое и правое поддеревья, а полученные слова после завершения функции выводятся.

Поиск глубины дерева происходит в функции «get_tree_depth», куда передается дерево. Происходит получение глубины левого поддерева и правого. Если текущая глубина левого больше, чем правого, то функция возвращает левую глубину, увеличенную на единицу (так как нужно учитывать корень дерева). Иначе – правую глубину, также увеличенную на единицу.

Запись дерева в файл происходит с помощью функции «write_to_file», куда передается файл, дерево, строка с сепараторами и флаг. В данной функции происходит запись слов в файл столько раз, чему равен их счетчик, а между словами ставится первый из сепараторов. Таким образом обходится левое и правое поддеревья. После записи в файл происходит восстановление дерева из файла с помощью функции «read_file», описанной ранее.

Печать дерева происходит с помощью функции «print_tree», куда передается дерево и уровень. В цикле происходит печать символов, которые разделяют узлы дерева для наглядности, а затем выводится слово и счетчик. Таким образом обрабатывается левое и правое поддеревья.

Задание №10. Построение дерева скобочного выражения

Условие: 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: в программе определена структура «Node», элементами которой является «data», где хранится считанный элемент, «count», который

содержит информацию о количестве потомков узла, ссылка на следующий элемент и указатель на массив указателей на дочерние узлы.

Сначала проверяются поданные имена файлов с помощью функции «check_name». Происходит посимвольная проверка на равенство имен. Если имена разные, то возвращается успех, иначе – ошибка.

После проверки и открытия файлов происходит построчная обработка входного файла. Для каждой строки вызывается функция «build_tree», которая строит дерево.

В функции в цикле происходит обработка всех элементов строки. Если элемент – открывающаяся скобка, то происходит добавление узла дерева из следующего за скобкой элемента, а также узел добавляется в массив указателей. Если элемент – запятая, то узел создается из элемента, который идет через один (чтобы пропустить пробел), также этот узел становится дочерним для текущей вершины. Если элемент – закрывающаяся скобка, то происходит получение последнего элемента из массива. Таким образом, массив предназначен для отслеживания вложенности скобок.

Функция «create_node» создает узел дерева. Переданный в функцию символ записывается в соответствующее поле «data». Остальные поля получаю значение «NULL».

Функция «add_child» добавляет потомков в дерево. Сначала увеличивается счетчик у родительского узла, а далее, при успешном выделении памяти, происходит добавление дочернего узла.

Функция «delete» удаляет дерево, проходя по нему и очищая использованную память.

Функция «print_tree» печатает дерево в файл, добавляя пробелы в зависимости от уровня, который был передан в функцию. Функция рекурсивно проходит по всему дереву и записывает узлы в файл.

Список заданий №4

Задание №1. Реализация хэш-таблицы

Условие: 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 допускается использование символов латинского алфавита (прописные и строчные буквы не отождествляются) и символов арабских цифр; значение 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 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 в рамках хеш-таблицы вычисляйте как остаток от деления эквивалентного для 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: в программе определена структура «Hash», элементами которой являются размер таблицы и указатель на массив указателей на элементы; структура «Elements», которая является двусвязным списком, полями которого являются указатели на первый и последний элементы списка; структура «Elem», которая является элементом списка, полями которой является макрос, макроподстановка, значение хэша, указатель на следующий и предыдущий элементы списка.

Первоначально происходит открытие входного файла, инициализация хэш-таблицы, списка.

Обработка файла происходит в функции «read_file», куда передается входной файл, его имя и адрес хэш-таблицы. В данной функции происходит получение имя файла для результата с помощью функции «get_file_name» (в данной функции происходит получение случайных символов, которые

записываются в строку. В конце строки добавляется «.txt», а затем строка с именем файла возвращается). После открытия файла для записи результата происходит построчная обработка файла с данными с помощью цикла «while». Выделяется память для хранения считанного макроса и для считанной макроподстановки. Если в начале строки содержится «#define», то происходит пропуск пробелов. Далее происходит запись символов из строки в строку макроса, до появления пробела. При необходимости происходит увеличение памяти для строки. После этого происходит проверка макроса с помощью функции «check_macro» (в этой функции происходит посимвольная проверка макроса; если в записи содержатся символы кроме цифр и букв, то возвращается ошибка, иначе – успех). Далее происходит аналогичное считывание макроподстановки. После этого вызывается функция «hash_function» для макроса.

В этой функции происходит перевод полученной строки в число. Происходит проход по строке справа налево и обрабатывается каждый символ строки. Если символ – это цифра, то из символа вычитается «0», если заглавная буква, то вычитается «A» и прибавляется 11, если строчная буква, то вычитается «a» и прибавляется 37. Прибавление чисел происходит для нахождения значения символа исходя из алфавита, который дан по условию. Полученная цифра умножается на степень и прибавляется к результату, а степень каждую итерацию умножается на 62. Итоговый результат возвращается.

После получения значения хэша происходит вставка элемента в таблицу с помощью функции «insert», куда передается таблица, значение хэша, макрос и макроподстановка.

В функции происходит получения значения индекса для вставки элемента путем нахождения остатка от деления хэша на текущий размер таблицы. Затем вызывается функция «insert_to_chain», куда передается место для вставки, значение хэша, макрос и макроподстановка.

Сначала в функции происходит проверка, содержится ли элемент уже в таблицы с помощью функции «find_in_chain», куда передается текущее место и макрос. В функции происходит проход по текущему списку (так как каждая ячейка в таблице — это список, а индекс ячейки уже был получен, то происходит проход по соответствующему списку) и проверка, содержится ли в списке уже переданный макрос. Если содержится, то из функции возвращается соответствующий элемент, иначе – «NULL». Если элемент содержится, то его записанная макроподстановка удаляется, а

вместо неё записывается новая, которая была получена. Иначе создается новый элемент и вставляется на место. Если в текущей ячейке еще не было элементов, то первый элементом в списке становится созданный. Иначе происходит вставка в конец списка.

После успешной вставки элемента происходит проверка таблицы с помощью функции «check_length», куда передается таблица. В данной функции происходит проход по всем ячейкам таблицы и проверка длины их списков. В результате прохода находится максимальная и минимальная длина. Если максимальная цепочка в два раза длиннее, чем минимальная, то необходимо пересобрать таблицу. Если в одной ячейке длина списка больше или равна 128, а другие ячейки – пустые, то пересборка таблицы не поможет (так как элементы после пересборки окажутся в той же ячейке), что может привести к ошибке работы программы, поэтому возвращается соответствующий статус-код, а программа завершается.

Пересборка таблицы происходит в функции «resize», куда передается таблица. Сначала происходит получение нового размера таблицы с помощью функции «get_size». В данной функции происходит нахождение следующего простого числа после текущего размера таблицы. Простота числа проверяется с помощью функции «check_prime», логика работы которой была описана ранее в других заданиях. После получения размера он присваивается новой таблице, выделяется память для элементов, инициализация списков в таблице. Далее в цикле происходит вычисление нового значения хэш-функции для каждого элемента старой таблицы, а затем вызывается функция «insert_to_chain» для вставки элемента в новую таблицу.

Очищение памяти, использованной для хэш-таблицы, происходит в функции «free_table», где в цикле очищаются все элементы таблицы.

После получения и добавления в таблицу всех макросов и макроподстановок происходит замена слов в тексте. Это происходит с помощью функции «replace_words», куда передается таблица, текущая строка текста и файл для записи результата. В данной функции происходит проход по строке текста. Определена строка для записи слова из текста. Если в строке встретились пробелы, то они записываются в итоговый файл. Если встретился символ, то он записывается в строку для слова до появления разделителя в строке. Происходит проверка, содержится ли слово в хэш-таблице в качестве макроса (с помощью функции «find_in_table»).

Если содержится, то происходит сохранение макроподстановки, а затем запись её в файл. Иначе в файл записывается исходное слово.

После обработки всех строк текста происходит закрытие всех файлов, очистка хэш-таблицы и завершение работы программы.

Задание №2. Обработка целочисленных массивов

Условие: 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 $[lb; lr]$ в 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: в программе определена структура «Array», элементами которой является символ, который является именем массива, указатель на массив с наполнением массива и размер массива.

Первоначально происходит открытие входного файла, а затем вызов функции «read_file», куда передается входной файл, объект структуры «Array», где будут храниться все массивы (размер равен 26, так как имена массивов могут быть только буквами латинского алфавита), счетчик для отслеживания числа массивов и имя входного файла.

В функции происходит построчная обработка файла. Происходит проход по строке до первого встреченного пробела (или разделителя). Если пробел не встречен или длина команды превышает 7, то строка пропускается. Иначе полученная из строки команда отправляется в функцию «command_check», откуда возвращается статус-код, соответствующий введенной команде. После получения команды происходит вызов функции «check_str» для проверки правильности введенной строки. В этой функции происходит проход по строке и проверка количества скобок, символов «;», пробелов в строке. Если всё правильно, то вызывается функция «change_command», куда передается команда, строка, число массивов, объект структуры, имя файла.

В самой функции происходит вызов функций в зависимости от полученной команды.

Команда «Load». С помощью функции «get_name», куда передается исходная строка с командой, флаг, место для записи результата и текущая позиция в строке, происходит получение имени массива и имени файла.

Функция «get_name» производит получение имени массива или имени файла из строки из входного файла. Обработка происходит в зависимости от переданного флага. Если флаг равняется 1, то происходит получение имени массива. Происходит проход до конца строки или до запятой, а предыдущий символ от разделителя будет именем массива. Он записывается в строку. Если это не буква, то возвращается ошибка. Иначе – текущая позиция в строке. Если флаг равняется 2, то происходит получение имени файла. Отличается от предыдущего случая тем, что здесь используется увеличение памяти при необходимости, так как символов может быть много, а в имени массива – только один. Если флаг равняется 3, то также происходит получение имени массива, но этот случай используется для команд, где 4 аргумента. Если флаг равняется 4, то происходит получение чисел из строки с командой.

После получения имени массива происходит создание массива с помощью функции «create». В этой функции происходит инициализация нового массива. Затем количество массивов в системе увеличивается, а массив помещается в общий массив. Также происходит сравнение полученного имени файла с именем входного файла (они не должны быть одинаковыми). После этого происходит вызов функции «load_cmd», куда передается массив и полученный файл. В функции осуществляется получение чисел из файла и сохранение их в массив. При необходимости происходит увеличение размера массива. После успешного завершения функции массив выводится на экран с помощью функции «print_array», куда передается массив и границы печати. В данной функции с помощью цикла происходит печать элементов, переданные границы указывают на то, какие элементы будут напечатаны.

Команда «Save». Происходит получение имени массива и имени файла. Далее происходит поиск массива среди сохраненных массивов с помощью функции «find_array», куда передается имя массива, массив с массивами и их количество. В функции происходит проход по всем массивам и сравнение их имени с переданным. Если найдено совпадение, то массив возвращается. Иначе возвращается «NULL». Если массив найден, то происходит вызов функции «save_cmd», куда передается массив и файл, иначе выводится соответствующее сообщение об ошибке. В функции происходит проход по массиву и печать всех элементов в файл.

Команда «Rand». Происходит получение имени массива, количества элементов, а также границ. Если все введено верно, то происходит вызов

функции «rand_cmd», куда передается количество элементов, границы и массив. В функции происходит обновление размера массива, а также заполнение его случайными элементами с помощью «rand()». Заполнение происходит аналогично предыдущим заданиям, где использовалось «rand()».

Команда «Concat». Происходит получение имен массивов, а также нахождение их в системе. Если все успешно, то происходит вызов функции «concat_cmd», куда передается два массива. В данной функции происходит обновление размера первого массива, а также запись элементов из второго массива в конец первого.

Команда «Free». Происходит получение имени массива и нахождение его среди сохраненных массивов. Если все успешно, то вызывается функция «free_cmd», куда передается массив. В функции происходит очищение массива, его размер приравнивается к нулю.

Команда «Remove». Аналогично предыдущим командам происходит получение всех элементов из входной строки. Если все верно и массив найден в системе, то вызывается функция «remove_cmd», куда передается массив, индекс и количество элементов. В функции в цикле происходит сдвиг элементов, чтобы исключить переданное количество элементов, затем уменьшение размера массива и его переопределение.

Команда «Copy». Обработка осуществляется с помощью функции «copy_cmd», куда передаются два массива и границы копирования. В функции происходит обновление размера итогового массива, а затем копирование элементов из первого массива во второй.

Команда «Sort». Обработка осуществляется функцией «sort_cmd», куда передается массив и «+» или «-», в зависимости от введенного. Сортировка осуществляется с помощью «qsort». Если передан «+», то в качестве компаратора используется функция «comparison_plus», иначе – «comparison_minus». Функции отличаются тем, что в «comparison_minus» возвращается результат умножается на -1, так как нужно отсортировать по убыванию.

Команда «Shuffle». Сортировка полученного массива происходит с помощью «qsort», в качестве компаратора используется функция «comparison_random». Эта функция возвращает случайное число из -1,0,1, что позволит переставить числа в массиве.

Команда «Stats». Обработка осуществляется функцией «stats_cmd», куда передается массив. Первоначально выводится размер и имя массива. Если размер равняется нулю, то все остальные пункты информации также будут нулевые. Иначе происходит получение значений. Максимальный элемент получается с помощью функции «get_max», куда передается массив. В функции осуществляется проход по массиву и сравнение каждого элемента с текущим максимумом. Найденный максимум возвращается. Минимальный элемент находится с помощью функции «get_min», куда передается массив. Нахождение происходит аналогично нахождению максимума. Нахождение наиболее частого элемента происходит с помощью функции «get_freq», куда передается массив. Осуществляется проход по массиву и для каждого элемента происходит также проход и подсчет количества его появлений в массиве. Если оно больше текущего максимума (или число равно, но значение элемента больше того элемента, который сейчас считается самым частым), то максимум обновляется. Найденный элемент возвращается. Нахождение среднего значения происходит с помощью функции «get_average», куда передается массив. В цикле осуществляется суммирование всех элементов массива, а полученная сумма возвращается поделенная на размер массива. Нахождение максимального отклонения происходит в функции «get_max_dev», куда передается массив и найденное среднее значение. В функции в цикле происходит нахождение модуля от разности текущего элемента и среднего значения. Если полученная разность больше максимума, то максимум обновляется. Найденное значение возвращается. После получения всех значений они выводятся.

Команда «Print». Обработка осуществляется функцией «print_array», которая была описана ранее.

Задание №5. Обработка арифметических выражений

Условие: 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: в программе определена структура «Stack», элементом которой является указатель на верхний элемент стека, а также структура «Node», элементами которой является значение, оператор и указатель на следующий элемент.

Обработка входных файлов происходит в цикле. Для каждого файла происходит вызов функции «read_file», куда передается файл, номер файла и его имя. В функции происходит открытие файла для записи ошибок (имя файла — это номер файла, который был подан на обработку). Далее в цикле происходит построчная обработка входного файла. Сначала происходит удаление «\n» из конца строки с помощью функции «remove_n». В этой функции данный символ заменяется на «\0». Далее вызывается функция «check_brackets» для проверки баланса скобок.

В данной функции осуществляется проход по строке с выражением. Если была встречена открывающаяся скобка, то счетчик увеличивается,

если закрывающаяся – то уменьшается. Если после обработки строки счетчик не равен нулю, то баланс нарушен, иначе все хорошо.

После проверки баланса скобок происходит построение обратной польской записи выражения. Это происходит с помощью функции «convert», куда передается строка и место для записи результата.

В самой функции создается стек с помощью функции «create_stack», где происходит инициализация. Далее в цикле происходит разбор выражения. Если в строке встречена буква или неправильная скобка, то стек очищается с помощью функции «free_stack» и возвращается ошибка. Если текущий символ — это цифра (или «-», то есть отрицательное число), то происходит получение числа, а затем запись числа в итоговую строку. Если текущий символ — это открывающаяся скобка, то она помещается в стек. Если это зарывающаяся скобка, то происходит выталкивание из стека записанных элементов, пока не будет встречена открывающаяся скобка или стек не опустеет. Полученные из стека операторы записываются в строку. Если текущий символ – это оператор, то происходит выталкивание элементов из стека, пока приоритет текущего оператора меньше того, который получен в стеке или пока стек не опустеет. Полученные из стека операторы также записываются в строку. После выталкивания всех операторов в стек помещается текущий символ. Если после обработки строки стек не пустой, то из него также все выталкивается и записывается в строку.

Приоритет операторов получается с помощью функции «priority», которая возвращает соответствующее число исходя из полученного оператора.

Элементы помещаются в стек с помощью функции «push». В данную функцию передается стек, число и оператор (в зависимости от того, что кладется в стек, один из аргументов будет нулевым). Происходит создание нового элемента, а затем заполнение полей и обновление верхушки стека.

Выталкивание элементов из стека происходит в функции «pop». В данную функцию передается стек, а также указатели на место для записи числа или оператора. Во временную переменную помещается верхний элемент стека. Поля этого элемента сохраняются в переданные указатели, верхушкой стека становится следующий элемент, а временный очищается.

Если функция «convert» возвращает ошибку, связанную с неверными символами в выражении, то ошибка записывается в файл. Иначе вызывается

функция «count», куда передается полученная строка и переменная для сохранения результата.

В функции «count» происходит получение результата выражения. Также создается стек, а строка обрабатывается в цикле. Если текущий символ — это число (или «-», если это первый символ в строке), то происходит получение числа, а далее оно помещается в стек. Если символ — это операция, то происходит получение операндов из стека, а затем вычисление операции. Полученный результат также помещается в стек. То есть в стек помещаются все операнды, а при операции они по двое выталкиваются для получения результата. После обработки строки результат выталкивается из стека.

Данная функция может вернуть ошибки при недостаточном количестве операндов, делении на ноль, отрицательной степени, неправильно записанном выражении. Все эти ошибки при появлении записываются в файл с ошибками.

Иначе, если все успешно подсчитано, полученные результаты выводятся на экран.

Задание №6. Построение таблицы истинности

Условие: 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: в программе объявлена структура «Node_stack», которая является стеком для узлов дерева и содержит указатель на вершину стека; структура «Node_stack_n», которая содержит указатель на узел дерева и указатель на следующий элемент; структура «Node», которая содержит поле для сохранения символа, указатель на левое и правое поддеревья; структура «Stack», элементом которой является указатель на верхний элемент стека, а также структура «Node», элементами которой является значение, оператор и указатель на следующий элемент.

Первоначально происходит открытие файла с входными данными, а затем этот файл передается в функцию «read_file».

В этой функции происходит получение строки из файла. Из строки удаляется «\n» с помощью функции «remove_n», которая была описана ранее. Также происходит проверка баланса скобок с помощью функции «check_brackets», которая была описана в предыдущем задании. С помощью функции «get_vars» происходит получение количества, а также имен переменных из выражения. В данной функции осуществляется проход по строке. Если текущий символ — это буква, а также он ещё не был записан в массив, то происходит изменение элемента массива с нуля на единицу. У массива размер 26 – равен количеству букв в латинском алфавите. Индекс элемента равняется разности символа и «A». После получения числа переменных происходит получение обратной польской записи с помощью функции «convert».

Эта функция аналогична функции получения обратной польской записи из прошлого задания. Отличие состоит в том, что символы букв записываются в результат, а цифры, кроме единицы и нуля, считаются

неверными и приводят к возвращению ошибки. 1 и 0 также записываются в строку. Вспомогательными функциями также являются «priority», которая возвращает приоритет операторов, «push», которая помещает элемент в массив, «pop», которая выталкивает элемент и «create_stack» - инициализация стека. «free_stack», соответственно, очищает стек.

После получения постфиксной записи выражения происходит построение дерева с помощью функции «build_tree». В этой функции используется стек для узлов дерева. Он инициализируется с помощью функции «create_node_stack». В цикле также осуществляется проход по строке аналогично функции «convert». Если была встречена буква или цифра (1 или 0), то происходит создание узла дерева с помощью функции «create_node», где узел инициализируется. Также происходит добавление узла в стек. Если получен оператор «~», то есть отрицание, то из стека выталкивается узел, создается узел операции, левым потомком которого становится вытолкнутый операнд. Затем полученный узел операции помещается в стек. Это сделано для того, чтобы не потерять операцию отрицания. Если текущий символ — это операция, то из стека выталкиваются два операнда, а из операции создается узел дерева. Левым и правым потомком созданного узла становятся операнды, а затем узел помещается в стек. Корнем построенного дерева будет узел, вытолкнутый из стека после обработки строки. Стек очищается функцией «free_node_stack».

Узлы помещаются в стек с помощью функции «push_node». В функции происходит выделение памяти для узла, а затем обновление поля узла «tree_node», куда помещается переданный в функцию узел, а также обновление верхушки стека.

Узлы выталкиваются из стека функцией «pop_node», которая возвращает верхний элемент из стека и меняет верх стека.

После построения дерева происходит печать таблицы истинности в файл с помощью функции «print_table». В функции происходит получение случайного имени файла с помощью функции «get_file_name», которая была описана в предыдущих заданиях. Происходит получение числа строк (это 2 в степени, равной числу переменных). Далее переменные записываются в файл, а затем в двух циклах происходит получение 0 или 1. Происходит сдвиг текущего номера ряда вправо на число байт, равному переменной. То есть происходит деление числа ряда на 2 в степени, равной текущему номеру переменной. Таким образом заполняется массив со

значениями для текущей переменной (получается, ряд таблицы заполняется 0 и 1), а значения печатаются в файл. Затем происходит вычисление результата операции с помощью функции «calculate», куда передается корень дерева, полученные значение, число переменных и их имена. Полученный результат печатается в файл.

В функции «calculate» сначала проверяется, чем является переданный узел дерева. Если это число, то оно возвращается как разность символа и 0. Если это переменная, то находится её значение в массиве значений, и оно возвращается. Если это операция, то происходит её вычисление. Таким образом обрабатывается левое и правое поддерево.

Дерево очищается с помощью функции «free_tree», где рекурсивно очищается левое и правое поддерево.

Задание №7. Реализация структуры MemoryCell

Условие: 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: в программе определена структура «MemoryCell», элементами которой является указатель на строку для хранения имени переменной и переменная для хранения значения.

Для чтения данных из файла используется функция «read_file», куда передается входной файл, массив структур для хранения переменных, размер структуры и количество переменных. В функции происходит построчное чтение входного файла. Выделяется память для хранения имени переменной и значения. Происходит сохранение всех символов строки, пока не был получен символ «=». Если было получено слово «print», то происходит либо получение имени переменной, которую нужно напечатать, либо происходит печать всех сохраненных переменных. Иначе продолжается разбор строки. Полученное имя переменной проверяется с помощью функции «check_name», где происходит проверка на то, что в имени содержатся только символы букв. Если после получения значения переменной в строке идет символ «;», то происходит сохранение значения. Иначе происходит получение второго значения и сохранения знака операции между переменными.

Печать всех элементов происходит с помощью функции «print_all_memory», куда передается указатель на элементы и их количество. В цикле происходит печать всех переменных и их значений.

Печать одного элемента происходит с помощью функции «print_elem», куда передается указатель на элементы, их число и имя переменной. Первоначально происходит проверка, содержится ли элемент в системе. Это происходит с помощью функции «find_var», куда передаются элементы, их количество и имя искомой переменной. Если переменная есть в системе, то происходит её печать.

Поиск в функции «find_var» осуществляется согласно алгоритму дихотомического поиска [11]. Создается переменная «left», изначальное значение которой равняется нулю, и «right», значение которой – правый конец массива, число сохраненных переменных, уменьшенное на единицу. Пока левая граница меньше правой происходит поиск по массиву в цикле. Находится середина массива и происходит сравнение имени найденной переменной и искомой. Сравнение происходит с помощью «strcmp», поэтому, если результат сравнения равен нулю, то переменная найдена и она возвращается. Иначе происходит сдвиг границы либо вправо, либо влево. Если переменная не найдена после прохода, то возвращается «NULL».

Сохранение значения переменной происходит с помощью функции «save», куда передается имя, значение, текущий размер массива переменных, сам массив, текущее число переменных и флаг, который указывает на то, является значение отрицательным или нет. Происходит проверка валидности значения с помощью функции «check_digit» (значение должно либо состоять только из букв, значит, нужно взять значение сохраненной переменной и передать новой, либо только из цифр, то есть мы записываем новое значение). Если значение состоит только из цифр, то оно записывается в переменную «num». Иначе происходит нахождение переменной в системе и её значение сохраняется в переменную «num». Затем происходит проверка, есть ли в системе уже переменная с переданным именем. Если это так, то у переменной просто меняется числовое значение. Иначе происходит проверка на то, достаточно ли памяти у массива. Если достаточно, то переменная просто сохраняется в конец массива, иначе происходит довыделение памяти. Далее массив сортируется с помощью функции «qsort», компаратором является функция «compare_names», которая возвращает результат сравнения имен переменных.

Если значение переменной должно быть результатом операции между двумя переменными, то это происходит с помощью функции «get_vars». Сначала происходит проверка первого значения с помощью функции «check_digit». Проверка и сохранение результата происходит аналогично функции «save». Также происходит получение второго значения. Результат операции между ними находится с помощью функции «change_op». В этой функции сначала происходит получение результата операции в зависимости от переданной операции, а затем получение из числового результата строки с помощью функции «get_str_result». В этой функции каждая цифра числа записывается в строку, а затем строка переворачивается, так как изначально число записывалось справа налево. Полученный строчный результат передается в функцию «save» для сохранения.

Очищение памяти, занятой массивом, происходит в функции «free_array», где в цикле очищаются все элементы.

Вывод

В данной курсовой работе реализовано 35 программ на языке Си для решения различных задач. В процессе решения изучен синтаксис языка, его особенности. В ходе работы использованы различные структуры данных, алгоритмы для поиска данных, выполнения расчетов, обработки входных данных, например бинарный поиск, быстрое возведение в степень, получение постфиксной записи выражения, соответственно. Многие задачи требовали работы с динамической памятью, указателями, так что данные темы также изучены.

Таким образом, получен большой опыт написания программ на языке Си, освоены многие инструменты данного языка.

Источники

1. [Электронный ресурс] Sum of Integers Formula // Cuemath URL: https://www.cuemath.com/sum-of-integers-formula/ (дата обращения: 25.09.2023).
2. [Электронный ресурс] Рекуррентная формула // Helpiks URL: https://helpiks.org/9-30748.html (дата обращения: 26.09.2023).
3. [Электронный ресурс] Рекурсивное вычисление факториала на Си // vscode URL: https://vscode.ru/prog-lessons/rekursivnoe-vyichislenie-faktoriala-na-si.html (дата обращения: 26.09.2023).
4. [Электронный ресурс] Метод Ньютона // Алгоритмика URL: https://ru.algorithmica.org/cs/numerical/newton/ (дата обращения: 26.09.2023).
5. [Электронный ресурс] Вычисление интеграла на Си // vscode URL: https://vscode.ru/prog-lessons/vyichislenie-integrala-na-si.html (дата обращения: 02.10.2023).
6. [Электронный ресурс] Генерация случайных чисел в языке Си // Young Coder URL: https://youngcoder.ru/lessons/4/sluchainie_chisla_na_c.php (дата обращения: 05.10.2023).
7. [Электронный ресурс] Бинарное возведение в степень // Алгоритмика URL: https://ru.algorithmica.org/cs/algebra/binpow/ (дата обращения: 25.10.2023).
8. [Электронный ресурс] Как проверить многоугольник на выпуклость // aurum42 URL: https://aurum42.ru/kak-proverit-mnogougolnik-na-vypuklost/ (дата обращения: 29.10.2023).
9. [Электронный ресурс] Простыми словами о методе дихотомии и золотом сечении: как найти корень уравнения без сложных вычислений // Научные статьи URL: https://nauchniestati.ru/spravka/metod-dihotomii-i-zolotogo-secheniya/ (дата обращения: 27.10.2023).
10. [Электронный ресурс] Разложение многочлена по степеням двучлена // StudFiles URL: https://studfile.net/preview/7757665/page:7/ (дата обращения: 29.10.2023).
11. [Электронный ресурс] Алгоритмы поиска в линейных структурах // ИНТУИТ URL: https://intuit.ru/studies/courses/648/504/lecture/11466 (дата обращения: 01.12.2023).

Приложение

Реализация всех описанных задач на github.com: https://github.com/adfxxx/Math_Prac