Общие указания

Поскольку данные лабораторные работы требуют знания только основ языка С, их можно выполнять с помощью любого компилятора С. В частности, удобными являются т.н. онлайн-компиляторы, не требующие установки на компьютер.
К примеру:
https://www.onlinegdb.com/online_c_compiler
https://www.programiz.com/c-programming/online-compiler/
- в первом из них лучше реализована работа с файлами.

Лабораторная работа № 1. Линейные алгоритмы

Полезная информация к работе.

Пример простой программы вычисления значения функции в точке, задаваемой пользователем:

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

void main()
{
double x;

printf("Calculation of function ln(x)\n");
printf("(x > 0) x = ");
scanf("%lg", &x);
printf("f(x) = %f\n", log(x));
}
```

Весь исполняемый код должен располагаться между открывающей фигурной скобкой после **void main()** и парной к ней закрывающей.

В языке С весь ввод – вывод информации, в том числе и общение с человеком (например, вывод информации на экран и ввод её с клавиатуры), производится только через обращение к функциям.

Функция форматного вывода на экран называется **printf.** Её первым аргументом должна быть строковая константа, называемая форматной строкой. Она определяет, сколько и в каком виде будет выведено на экран значений переменных. Остальными аргументами функции являются те переменные (или выражения), значения которых надо вывести на экран. Пример: пусть мы хотим вывести на экран значение переменной целого типа **a**, при этом отметив, что значение **a** является результатом вычислений. Тогда мы можем написать

printf(“result=%d and that is good!”, a);

В результате выполнения этого оператора, если текущее значение **a** было равно 5, на экран будет выведено:

result=5 and that is good!

Нетрудно видеть, что на экран выводится форматная строка, в которую в нужных местах вставляются значения выводимых переменных. В данном случае надо было вывести целое значение, и нужное место было помечено форматом вывода целого значения **%d**. Для вывода вещественного значения типа **float** нужное место надо отметить символами **%g ,**

для вещественного типа **double** (двойной точности) - **%lg** , для символьного - **%c** , и для строкового - **%s**. Пример:

printf("Variable x is %g\n and variable y is %c", x, y);

- если **x** это вещественная переменная и её значение равно **7.538**, а **y** – символьная переменная и её значением является символ **A**, то при своём исполнении функция **printf** выведет на экран

Variable x is 7.538
and variable y is A

Нельзя путать форматы! Если вы по ошибке попробуете вывести значение не соответствующим его типу форматом, значение на экране будет неправильным, как правило, ничего общего с реальным значением не имеющим!

Функция форматного ввода с клавиатуры называется **scanf** и работать с ней надо так: если нам надо ввести с клавиатуры значение переменной **x** вещественного типа **float**, то в нужном месте программы надо написать

scanf("%g", &x);

Начав выполнять это выражение, программа остановится и будет ждать, пока мы не введём с клавиатуры вещественную константу, завершив ввод нажатием кнопки Enter. После этого введённое значение будет присвоено переменной **x.**

Выше в примере простой программы для ввода значения типа **double** был использован формат **%lg**.

Обратите внимание на три важных обстоятельства: во-первых, форматы ввода, такие же, как и форматы вывода, нельзя путать. Если спутаете, значение, присвоенное переменной, будет неправильным. Во-вторых, перед именами вводимых переменных надо обязательно ставить знак **&**. Если этого не сделать, то работа программы может стать непредсказуемой! Но, в-третьих, из этого важного правила есть важное же исключение – если вводится значение строковой переменной по формату **%s**, перед её именем в аргументах функции **scanf** ставить знак **&** не надо. Иначе работа программы может стать непредсказуемой!

Когда при выполнении задания к лабораторной работе понадобятся названия математических функций в стандартной библиотеке языка С, их нужно искать в описании заголовочного файла **math.h**.

Требования по оформлению программы

При запуске на исполнение на экран должна выводиться следующая информация:

```
Лабораторная работа № 1

№ варианта, группа, автор (фамилия и имя полностью)

Задание1.
```

Далее текст задания. Потом ввод данных для расчётов и вывод на экран результатов расчётов для задания 1.

Потом тоже самое для задания 2, задания 3 и т.д.

Контрольные вопросы

1. Перечислить все знаки операций в С с особенностями их выполнения.
2. Перечислить элементарные типы данных в С с описанием их размера и формата.
3. Рассказать об использовании функции **printf**.
4. Рассказать об использовании функции **scanf**.

Лабораторная работа № 2. Простые циклы и условные операторы

Полезная информация к работе.

1). Заполнение массива случайными значениями на языке С может быть произведена при помощи функции **rand()**, объявленной в заголовочном файле **stdlib.h**. Она возвращает случайное целое число в диапазоне от **0** до **RAND_MAX** – специальной именованной константы, равной максимальному целому числу. Если надо ограничить диапазон случайного числа некоторым максимальным значением **K**, это можно сделать, взяв остаток от деления: **rand()%(K+1)**.

Требования по оформлению программы

Аналогичны требованиям лабораторной работы №1.

Контрольные вопросы

1. Рассказать о синтаксисе условных операторов С – **if** и **switch**.
2. Рассказать о синтаксисе операторов цикла в С – **while, do-while** и **for**.
3. Рассказать о назначении и синтаксисе вспомогательных операторов – **break** и **continue**.
4. Рассказать о синтаксисе определения функций в С.

Лабораторная работа № 3. Работа с файлами.

Полезная информация к работе.

1). Функции работы с символьными данными в стандартной библиотеке С описываются в файле **ctype.h**. Например,

int isalpha(int ch) – проверка, является ли символ буквой;

int isdigit(int ch) – проверка, является ли символ цифрой;

int toupper(int ch) – если аргумент – строчная буква, то переводит её в прописные, все остальные символы оставляет без изменений. И ещё много других полезных функций.

2). Для сбора необходимой при решении задач данной лабораторной работы информации, очень удобной структурой данных является таблица, ключ в которой может быть получен из ASCII кода прочитанного из файла символа. Например, если нужно собрать информацию о встречающихся в файле латинских буквах, то можно использовать, то что в таблице кодов ASCII они идут подряд в алфавитном порядке – сначала большие, потом, через некоторый промежуток, маленькие. То есть, выражение (**ch – ‘A’)**, где **ch** - некоторая латинская заглавная буква, будет равно **0** для **ch==’A’**, 1 для **ch==’B’** и т.д. до 25 для **ch==’Z’ –** и может являться индексом в массиве целых чисел (размером 26)**.** А сам массив может хранить информацию как о просто наличии этой буквы, так и о количестве этих букв в строке или файле. То же самое и для цифр.

3). Также удобным инструментом для такого рода задач является посимвольный ввод-вывод функциями **fgetc** и **fputc**.

Требования по оформлению программы

Аналогичны требованиям лабораторной работы №1, кроме вывода на экран текста задания – текст задания выводить на экран не надо.

Контрольные вопросы

1. Рассказать о функциях работы с файлами.
2. Рассказать о директивах препроцессора С: **#include**, **#define**.
3. Рассказать о кодировке символов в С.
4. Рассказать о синтаксисе массивов в С и работе с ними.

Лабораторная работа № 4. Некоторые полезные приёмы математических вычислений.

Полезная информация к работе

1). ***Рекурсия*** – есть метод определения множества объектов или процесса в терминах самого себя.

Любое рекурсивное определение содержит две части:
- базисную часть
- Рекурсивную часть.
Базисная часть (условие завершения (одно или несколько)) является **нерекурсивным** утверждением, которое может быть вычислено для определенных параметров. Таким образом, базисная часть может задавать один или более случаев остановки рекурсии.
Рекурсивная часть определения записывается таким образом, чтобы при цепочке повторных применений утверждение из рекурсивной части приводилось бы к базисной части.
Рекурсивные алгоритмы реализуются через ***рекурсивные функции***

Прямая (явная) рекурсия характеризуется наличием в теле процедуры оператора обращения к ней же самой.
В случае ***косвенной (неявной) рекурсии*** одна процедура обращается к другой, которая (возможно через цепочку вызовов других процедур) вновь обращается к первой.
В данной работе рассматривается только прямая рекурсия.

Фрейм активации: так как обращаться к рекурсивной процедуре можно как из нее самой, так и извне, каждое обращение к рекурсивной процедуре вызывает ее **независимую активацию.**
При каждой активации **образуются копии всех локальных переменных и формальных параметров рекурсивной процедуры**, которые используют операторы текущей активации.
Таким образом, для рекурсивной процедуры может одновременно существовать несколько активаций.
Для обеспечения правильного функционирования рекурсивной процедуры **необходимо сохранять адреса возврата** в таком порядке, чтобы возврат после завершения каждой текущей активации выполнялся в точку, соответствующую оператору, непосредственно следующему за оператором рекурсивного вызова.

- Совокупность локальных переменных, значений фактических параметров, подставляемых на место формальных параметров рекурсивной процедуры, и адреса возврата однозначно характеризует текущую активацию и образует ***фрейм активации***.

Фрейм активации необходимо сохранять при очередной активации и восстанавливать после завершения текущей активации. Так как при выходе из текущей активации самым первым должен быть восстановлен фрейм, который был позже всех сохранен, для хранения фреймов, т.н. область системного стека. Поэтому рекурсивные функции весьма затратные с точки зрения используемой памяти, и если не рекурсивная версия функции не слишком сложна по сравнению с рекурсивной версией, то лучше предпочесть её.

2). Суммирование рядов с рекуррентным вычислением членов ряда.

Ряд (бесконе́чная су́мма) — одно из центральных понятий математического анализа, математическая концепция, представляющая собой сумму бесконечного числа слагаемых, упорядоченных в определённой последовательности. Обозначение ряда:

$$\sum_{n=1}^{\infty} a_n = a_1 + a_2 + \cdots$$

Нумерация членов ряда не обязательно начинается с 1, часто с 0, но может и с другого целого числа. Если суммирование обрывается на N-том члене, получается ***частичная (конечная) сумма ряда***:

$$\sum_{n=1}^{N} a_n = a_1 + a_2 + \cdots + a_N$$

Если у последовательности частичных сумм существует предел, он называется ***суммой ряда***, а сам ряд называется ***сходящимся***, в противном случае – ***расходящимся***. Не всегда сумму ряда можно найти аналитически, в этом случае можно оценить её численно. Принято говорить при этом о численном суммировании рядов.

При численном суммировании рядов часто бывает эффективней (то есть, с меньшим числом операций и, соответственно, за меньшее время) не пользоваться функциями для возведения в степень или вычисления факториала, часто входящими в формулу для члена ряда, а вычислять члены ряда рекуррентно, то есть, следующий через предыдущий.

Пример: пусть надо посчитать сумму N членов ряда

$$S_N(x) = \sum_{n=0}^{N} \frac{x^n}{n!\,(2n+1)}$$

Чтобы получить зависимость n-го слагаемого от (n-1)-го, разделим одно на другое:

$$\frac{a_n}{a_{n-1}} = \frac{\dfrac{x^n}{n!\,(2n+1)}}{\dfrac{x^{n-1}}{(n-1)!\,(2(n-1)+1)}} = \frac{x}{n} \cdot \frac{2n-1}{2n+1}$$

Тогда нахождение нужной суммы с рекуррентным вычислением слагаемых на С можно записать, например, так:

```
#define N 100
…
float x;
…
float an=1,, Sn=1;              // это для n=0, когда a0=1 и сумма ряда из одного
                                // элемента (с индексом 0) S0=1
for(int n=1; n<=N; n++) {       // суммируем дальше, начиная с n=1
        an*= x*(2*n-1)/(n*(2n+1));
        Sn+= an;
}
```

Для справки: двойной факториал от чётного аргумента **(2n)!!** определяется, как произведение всех чётных чисел от **2** до **2n**. Двойной факториал от нечётного аргумента **(2n+1)!!** Определяется, как произведение всех нечётных чисел от **1** до **2n+1**. Двойные факториалы тоже часто попадаются в формулах для членов ряда.

Требования по оформлению программы

Аналогичны требованиям лабораторной работы №1, кроме вывода на экран текста задания – текст задания выводить на экран не надо.

Контрольные вопросы

1. Рассказать о преимуществах и недостатках рекурсии.
2. Рассказать о рекуррентных вычислениях.
3. Рассказать об указателях в языке С и работе с ними.
4. Рассказать о связи массивов с указателями в С.