##### ЭТАП Реализация программы

##### **Задание**

Разработать и отладить предварительную версию программы. Требования к программной реализации приведены в прикрепленном файле.

На данном этапе в первую очередь проверяется понятность и качество кода (дублирование кода, использование конструкций языка С, работа с памятью), а также соответствие между реализованными функциями и требованиями в постановке задачи.

При проверке руководителем могут быть указаны пропущенные требования, которые не вошли в постановку задачи или не были точно специфицированы, но текущая реализация этих требований делает интерфейс неудобным для использования. Также руководителем могут быть выявлены ошибки во внешнем поведении программы. Эти замечания не влияют на оценку за данный этап курсовой работы, но должны быть исправлены на этапе «[Тестирование и исправление ошибок в программе](https://edu.susu.ru/mod/assign/view.php?id=5873567)».

Критерии оценки

Общий балл за этап складывается из следующих показателей:

– понятность кода – есть комментарии ко всем функциям и глобальным переменным, используются именованные константы – 5 баллов, оценка снижается на 1 балл за каждую ошибку;

– качество кода – нет дублирования кода, слишком длинных функций, отсутствуют ошибки в применении конструкций языка С, нет необоснованных усложнений – 5 баллов, оценка снижается на 1 балл за каждую ошибку;

– полнота реализации требований – 5 баллов, оценка снижается на 1 балл за каждую нереализованную функцию из постановки задачи.

Максимальная оценка за этап – 15 баллов.

Программы должны составляться таким образом, чтобы их могли прочитать в первую очередь люди, а не машины. Программа — это документ для последующего использования.

Должен быть комментарий к каждой функции, глобальной переменной или структуре данных, объясняющий её назначение. Пояснительными комментариями нужно сопровождать те части программы, которые трудно понять без комментариев. Такой комментарий должен не переводить код на языке С на естественный язык, а содержать дополнительную информацию:

if(d<0) /\* корней нет \*/

В среднем на 10 строк программы приходится один такой комментарий. Комментарии к каждой строке кода могут указывать на чужое авторство программы. Вы должны уметь объяснять действия в программе, основываясь только на коде, именах функций и переменных, а также комментариях к ним.

Логические блоки кода от 5 до 20 строк рекомендуется отделять пустыми строками, для такого блока можно сделать поясняющий комментарий, который поможет понять код, не выполняя анализ каждого оператора в блоке. Такой блок может соответствовать одному символу в схеме алгоритма.

Также не забывайте об отступах, показывающих структуру программу. Программа должна быть приятна для глаза. В состав MinIDE включена утилита для форматирования кода программы с отступами.

Правильный выбор имен — это залог удобочитаемости программ. Имена функций должны быть связаны с глаголами или отглагольными существительными. Для имен можно использовать сокращения от слов английского языка или русского языка, записанных транслитом. Глобальные имена должными быть достаточно длинными, имена локальных переменных можно делать короткими — от 1 до 3 символов. Имена констант нужно писать полностью прописными буквами.

Желательно, чтобы длина функции не превышала 50 строк, если получается больше — нужно выделить вспомогательные функции.

При написании кода нужно обращать внимание на повторяющиеся части кода. Рекомендуется выделить функцию или написать цикл по массиву данных, которыми эти части различаются. Вместо массива можно попытаться вычислить различающиеся данные по формуле.

Кроме повторения кода следует обращать внимание на повторение числовых и строковых констант. Таким константам нужно дать имена с помощью #define или enum. Рекомендуется использовать символьные константы вместо ASCII‑кодов символов, например, '0' вместо 48.

В простейшем случае весь код программы можно записать в одном файле. Если количество строк кода в программе превышает 200-400, то можно разбить программу на несколько модулей. В заголовочном файле записываются заголовки функций, объявления структур и глобальных переменных. Заголовочный файл подключается ко всем модулям, которые используют эти функции и переменные, а также к самому модулю для контроля корректности объявлений. Рассмотрим пример разбиения программы на модули.

***Файл module1.h***:

extern int size;

double square(double x);

***Файл module1.c***:

#include "module1.h"

int size=100;

double square(double x)

{ return x\*x;

}

***Файл module2.c***:

#include "module1.h"

int main()

{ double a=square(2.5);

++size;

}

В файле локальных настроек для программы, состоящей из нескольких модулей, в MinIDE нужно добавить строку:

build.goal=all

Результатом данного этапа является предварительная версия программы (альфа-версия). Проверяется в первую очередь понятность и качество кода, соответствие между реализованными функциями и требованиями в постановке задачи.

Руководителем могут быть указаны пропущенные требования, которые не вошли в постановку задачи или не были точно специфицированы, но текущая их реализация делает интерфейс неудобным для использования. Также руководителем могут быть выявлены ошибки во внешнем поведении программы, не соответствующие поставленной задаче. Эти замечания не влияют на оценку за данный этап, но должны быть исправлены на следующем этапе.