**Приложение №7**

|  |  |
| --- | --- |
| **Дата** | 16.12.2023 |
| **Тема** | 1. Разработка программы по выделенным программным модулям  2. Выполнение индивидуального задания варианта 10 (9) |
| **Задания** | 1. Разработать программу по выделенным программным модулям  2. Выполнение индивидуального задания варианта 10 (9) |

**1. Разработка программы по выделенным программным модулям**

В качестве модульной структуры программы принято использовать древовидную структуру, включая деревья со сросшимися ветвями. В узлах дерева размещаются модули, а стрелки показывают подчиненность модулей (т.е. в тексте модуля, из которого она исходит, имеется ссылка на модуль, в который она входит)

Модульная структура программы должна включать и совокупность спецификаций модулей, образующих эту программу. *Спецификация*программного модуля содержит:

* синтаксическую спецификацию его входов, позволяющую построить обращение к нему на используемом языке программирования
* функциональную спецификацию модуля (описание всех функций, выполняемых этим модулем).

Существуют разные методы разработки структуры программы. Обычно используют 2 метода:

* Метод восходящей разработки
* метод нисходящей разработки

Метод восходящей разработки. Сначала строится модульная структура программы в виде дерева. Затем поочередно программируются модули программы, начиная с модулей самого нижнего уровня, в таком порядке, чтобы для каждого программируемого модуля были уже запрограммированы все модули, к которым он может обращаться. После того, как все модули программы запрограммированы, производится их поочередное тестирование и отладка в принципе в таком же (восходящем) порядке, в каком велось их программирование.

На первый взгляд такой порядок разработки программы кажется вполне естественным: каждый модуль при программировании выражается через уже запрограммированные непосредственно подчиненные модули, а при тестировании использует уже отлаженные модули. Не рекомендуется, т.к.:

* для программирования какого-либо модуля совсем не требуется текстов используемых им модулей - для этого достаточно, чтобы каждый используемый модуль был лишь специфицирован (в объеме, позволяющем построить правильное обращение к нему), а для тестирования его возможно используемые модули заменять их имитаторами (заглушками).
* каждая программа в какой-то степени подчиняется некоторым внутренним для нее, но глобальным для ее модулей соображениям (принципам реализации, предположениям, структурам данных и т.п.), что определяет ее концептуальную целостность и формируется в процессе ее разработки. При восходящей разработке эта глобальная информация для модулей нижних уровней еще не ясна в полном объеме, поэтому очень часто приходится их перепрограммировать.
* при восходящем тестировании для каждого модуля (кроме головного) приходится создавать ведущую программу (модуль), которая должна подготовить для тестируемого модуля необходимое состояние информационной среды и произвести требуемое обращение к нему. Это приводит к большому объему "отладочного" программирования и в то же время не дает никакой гарантии, что тестирование модулей производилось именно в тех условиях, в которых они будут выполняться в рабочей программе.

Метод *нисходящей разработки*.

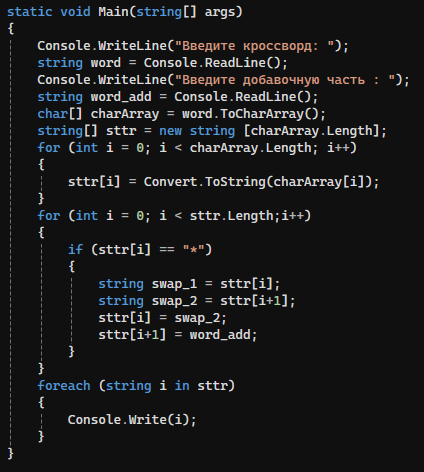
Сначала строится модульная структура программы в виде дерева. Затем поочередно программируются модули программы, начиная с модуля самого верхнего уровня (головного), переходя к программированию какого-либо другого модуля только в том случае, если уже запрограммирован модуль, который к нему обращается. После того, как все модули программы запрограммированы, производится их поочередное тестирование и отладка в таком же (нисходящем) порядке.

Положительные стороны

При таком порядке разработки программы вся необходимая глобальная информация формируется своевременно, т.е. ликвидируется весьма неприятный источник просчетов при программировании модулей.

Существенно облегчается и тестирование модулей, производимое при нисходящем тестировании программы. Первым тестируется головной модуль программы, который представляет всю тестируемую программу и поэтому тестируется при "естественном" состоянии информационной среды, при котором начинает выполняться эта программа. При этом все модули, к которым может обращаться головной, заменяются на их имитаторы. И*митатор модуля -*простой программный фрагмент, сигнализирующий, о самом факте обращения к имитируемому модулю с необходимой для правильной работы программы обработкой значений его входных параметров и с выдачей, если это необходимо, заранее запасенного подходящего результата. После завершения тестирования и отладки головного и любого последующего модуля производится переход к тестированию одного из модулей, которые в данный момент представлены имитаторами, если таковые имеются. Для этого имитатор выбранного для тестирования модуля заменяется на сам этот модуль и добавляются имитаторы тех модулей, к которым может обращаться выбранный для тестирования модуль. При этом каждый такой модуль будет тестироваться при "естественных" состояниях информационной среды, возникающих к моменту обращения к этому модулю при выполнении тестируемой программы. Таким образом большой объем "отладочного" программирования заменяется программированием достаточно простых имитаторов используемых в программе модулей. Кроме того, имитаторы удобно использовать для подыгрывания процессу подбора тестов путем задания нужных результатов, выдаваемых имитаторами.

**2. Выполнение индивидуального задания варианта 10 (9):**



**Выводы:** Разработал программу по выделенным программным модулям. Выполнил индивидуальное задание варианта 10 (9).

Студент: / Ульянов Никита Анатольевич