**Приложение №5**

|  |  |
| --- | --- |
| **Дата** | 13.12.2023 |
| **Тема** | 1. Анализ процесса обработки информации и выбор структур данных для её хранения.  2. Разработка основных алгоритмов решения задачи.  3. Выделение программных модулей для решения задачи.  4. Решения индивидуальных заданий варианта 10 (5-7) |
| **Задания** | 1. Проанализировать процесс обработки информации и выбор структур данных для её хранения. 2. Разработать основных алгоритмов решения задачи. 3. Выделение программных модулей для решения задачи. 4. Решить индивидуальные задания варианта 10 (5-7). |

**1. Проанализировать процесс обработки информации и выбор структур данных для её хранения:**

Система управления базами данных - объект для управления базами данных, отвечает за обособление данных и их дальнейшее хранение и обработку в разное время в неопределенном месте на сервере..

СУБД общается с БД, например с помощью языка SQL (Structured Query Language). Реляционная модель - публикация базы данных в виде таблиц с действиями для записей на языке SQL. Данный язык более абстрактный, чем C и остальные представленные на рынке, т.к. запрос остается на выбор администратора СУБД. Раньше данные хранились в алфавитно-цифровой форме, разделялись по стандартным типам. А теперь сюда включаются и бинарные объекты такие как изображения, видео и большие текста, по которым может происходить поиск и вывод информации.

Следующий очень нужным элементом СУБД является встроенный язык программирования, служит для автоматизации процессов обслуживания системы и анализа и обработки данных внутри базы. Приложения пользователя взаимодействуют с СУБД в двух или трех уровневой клиент-серверной архитектуре. От сюда следует, что физический сервер на котором установлена СУБД называется сервером баз данных. Управление и администрирование СУБД включает в себя создание базы данных, управление и обслуживание всей инфраструктуры сервера.

При выборе СУБДбудем отталкиваться от тех приложений, которыми она будет администрироваться, т.е. выбор СУБД - 100% задача разработчика, а не пользователя, который может даже не понимать что это такое. Ведущие поставщики на рынке СУБД: IBM, Oracle и Microsoft. При выборе необходимо руководствоваться такими параметрами, как масштабируемость, быстрота, работа с XML и другими кластерными решениями.

В данный момент времени все эти системы IBM, DB/2, MS SQL и Oracle примерно одинаковы по большинству ключевых параметров. На общем фоне всех выделяются только Cache из-за новизны подхода и особой идеологии архитектуры реализации.

Масштабность системы. Чем больше будет данных в базе данных, тем сложнее ими управлять, что логично.

Мультиплатформенность. Также реализуют разные возможности масштабирования системы, благодаря новым технологиям можно менять аппаратную составляющую системы и операционную систему на более подходящую в связи с растущими потребностями бизнеса, даже без потерь данных, что очень важно в данное время, но и так же смены прикладного ПО и переподготовки администратора базы данных.

Благодаря кластерным возможностям системы, в приложении к СУБД, например, по средствам Oracle RAC, эта технология повышает надежность всей системы, дает упрощение масштабируемости и снимает риски на развитие инфраструктуры в целом.

Различные системы имеют разные характерные черты. Например, IBM DB/2 имеет свою высокопроизводительную кластерную организацию, которая дает возможность переходить от разного рода больших RISC-серверов в качестве серверов базы данных к мейнфреймам. Oracle поддерживает XML DB. Oracle и IBM DB/2 поддерживают такую возможность как SQLJ, что особенно важно в нынешних архитектурах.

**2. Разработать основной алгоритм решения задачи:**

Исключительно важно использовать язык блок-схем при разработке алгоритма решения задачи. Решение одной и той же задачи может быть реализовано с помощью различных алгоритмов, отличающихся друг от друга как по времени счета и объему вычислений, так и по своей сложности. Запись этих алгоритмов с помощью блок-схем позволяет сравнивать их, выбирать наилучший алгоритм, упрощать, находить и устранять ошибки.

Отказ от языка блок-схем при разработке алгоритма и разработка алгоритма сразу на языке программирования приводит к значительным потерям времени, к выбору неоптимального алгоритма. Поэтому необходимо изначально разработать алгоритм решения задачи на языке блок-схем, после чего алгоритм перевести на язык программирования.

При разработке алгоритма сложной задачи используется метод пошаговой детализации. На первом шаге продумывается общая структура алгоритма без детальной проработки отдельных его частей. Блоки, требующие детализации, обводятся пунктирной линией и на последующих шагах разработки алгоритма продумываются и детализируются.

В процессе разработки алгоритма решения задачи можно выделить следующие этапы:

* Этап 1. Математическое описание решения задачи.
* Этап 2. Определение входных и выходных данных.
* Этап 3. Разработка алгоритма решения задачи.

Базовые алгоритмические конструкции

В теории программирования доказано, что для записи любого, сколь угодно сложного алгоритма достаточно трех базовых структур:

* следование (линейный алгоритм);
* ветвление (разветвляющийся алгоритм);
* цикл-пока (циклический алгоритм).

Линейные алгоритмы

Линейный алгоритм образуется из последовательности действий, следующих одно за другим. Например, для определения площади прямоугольника необходимо сначала задать длину первой стороны, затем задать длину второй стороны, а уже затем по формуле вычислить его площадь.

### Разветвляющиеся алгоритмы

Алгоритм ветвления содержит условие, в зависимости от которого выполняется та или иная последовательность действий.

### ****Циклические алгоритмы****

Циклический алгоритм **–**определяет повторение некоторой части действий (операций), пока не будет нарушено условие, выполнение которого проверяется в начале цикла. Совокупность операций, выполняемых многократно, называется телом цикла

**3. Выделение программных модулей для решения задачи:**

Модульное программирование является особым способом разработки программы, которая строится при этом из нескольких относительно независимых друг от друга частей – модулей. Понятие модуля является одним из центральных при разработке программного обеспечения.

**Модуль** — функционально законченный фрагмент программы, оформленный в виде отдельного файла с исходным кодом или поименованной непрерывной её части, предназначенный для использования в других программах. Модули позволяют разбивать сложные задачи на более мелкие в соответствии с принципом модульности. Обычно проектируются таким образом, чтобы предоставлять программистам удобную для многократного использования функциональность (интерфейс) в виде набора функций, классов, констант. Модули могут объединяться в пакеты и, далее, в библиотеки. Удобство использования модульной архитектуры заключается в возможности обновления (замены) модуля, без необходимости изменения остальной системы. В большинстве случаев различные модули могут запускаться как на одном сервере, так и на разных, для распределения нагрузки и создания распределенной архитектуры.

**Модульное программирование** — это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определенным правилам. Использование модульного программирования позволяет упростить тестирование программы и обнаружение ошибок. Аппаратно-зависимые подзадачи могут быть строго отделены от других подзадач, что улучшает мобильность создаваемых программ.

Суть **модульного программирования** состоит в разбиении сложной задачи на некоторое число более простых подзадач и составлении программ для решения достаточно независимо друг от друга. Модульность является одним из основных принципов построения программных проектов.

Языки программирования, поддерживающие модульный подход, описывают модуль как программную единицу, состоящую из двух основных частей - спецификации (интерфейса) и реализации. В спецификации приводятся такие характеристики объектов модуля, которые необходимы и достаточны для использования этих объектов в других модулях и программах. Это позволяет использовать объекты модулей только на основе информации об их интерфейсе (не ожидая их полного описания). В реализационной части модуля описывается представление и алгоритмы обработки, связанные с теми или иными объектами модуля.

Модуль является одним из средств, облегчающих верификацию программ. Модуль, как средство создания абстракции, выделяет спецификацию и локализует сведения о реализации.

Модули служат также целям создания проблемно-ориентированного контекста и локализации машинной зависимости.

**Концепция модульного программирования**

Концепцию модульного программирования можно сформулировать в виде нескольких понятий и положений:

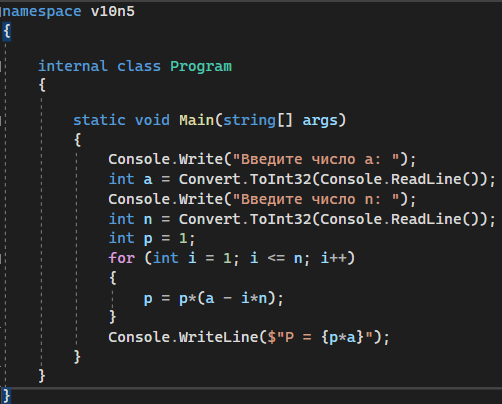
·Функциональная декомпозиция задачи- разбиение большой задачи на ряд более мелких, функционально самостоятельных подзадач - модулей. Модули связаны между собой только по входным и выходным данным.

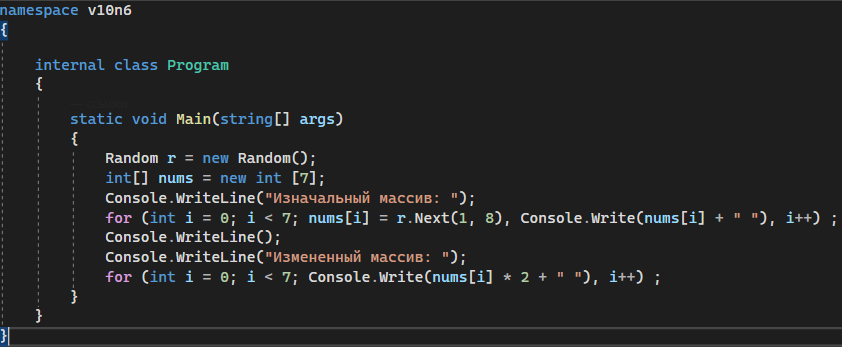
·Модуль - основа концепции модульного программирования. Каждый модуль в функциональной декомпозиции представляет собой "черный ящик" с одним входом и одним выходом. Модульный подход позволяет безболезненно производить модернизацию программы в процессе ее эксплуатации и облегчает ее сопровождение. Дополнительно модульный подход позволяет разрабатывать части программ одного проекта на разных языках программирования, после чего с помощью компоновочных средств объединять их в единый загрузочный модуль.

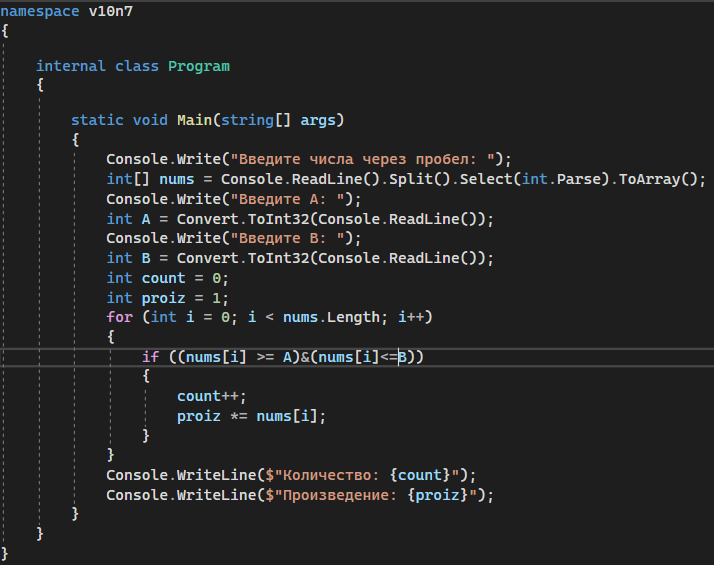
·Реализуемые решения должны быть простыми и ясными. Если назначение модуля непонятно, то это говорит о том, что декомпозиция начальной или промежуточной задачи была проведена недостаточно качественно. В этом случае необходимо еще раз проанализировать задачу и, возможно, провести дополнительное разбиение на подзадачи. При наличии сложных мест в проекте, их нужно подробнее документировать с помощью продуманной системы комментариев. Этот процесс нужно продолжать до тех пор, пока действительно не удастся добиться ясного понимания назначения всех модулей задачи и их оптимального сочетания.

Назначение всех переменных модуля должно быть описано с помощью комментариев по мере их определения.

**4. Решить индивидуальные задания варианта 10 (5-7):**







**Выводы:** Проанализировал процесс обработки информации и выбор структур данных для её хранения. Разработать основных алгоритмов решения задачи. Выделил программные модули для решения задачи. Решил индивидуальные задания варианта 10 (5-7).

Студент: / Ульянов Никита Анатольевич