***Экзаменационные вопросы по курсу «основы программирования»***

1. **Основные стадии разработки программного продукта.**

При разработке программного продукта можно выделить следующие стадии:

1. Стадия предпроектных исследований и технического задания (постановка задачи) — определение требований к программному продукту и осуществление формальной постановки задачи;

2. Стадия технического предложения (анализ) — определение методов решения задачи;

3. Стадия эскизного проектирования(проектирование) — разработка структуры программного продукта, выбор структур для хранения данных, построение и оценка алгоритмов подпрограмм, и определение особенностей взаимодействия программы с вычислительной средой (другими программами, операционной системой и техническими средствами);

4. Стадия технического проектирования (программирование) — составление программы на выбранном языке программирования, ее отладка и тестирование;

5. Стадия рабочего проектирования — оформление документации;

6. Стадии испытаний и внедрения в эксплуатацию — всестороннее тестирование программы и сопровождение при внедрении в эксплуатацию.

1. **Стадии предпроектных исследований и технического задания.**

Часто стадию предпроектных исследований и технического задания называют стадией постановки задачи. Постановка задачи должна содержать достаточно информации для того, чтобы позволить программисту или аналитику однозначно определить, что будет делать создаваемая программа. Детали того, как она будет это делать, должны определиться позже, при разработке алгоритма. Для того, чтобы грамотно выполнить постановку задачи, необходимо произвести предпроектные исследования предметной области, определить, существуют ли аналоги будущего программного продукта, их достоинства, недостатки и т.д.

Постановка решаемой на компьютере задачи должна включать список ее входных данных, список требуемых результатов и любые инструкции (правила), которых нужно следовать при решении задачи. В постановке также должно быть указано, должна ли задача решаться для конкретного случая (конкретных данных) или должна обобщаться для переменных входных данных. Если возможно, решение должно обобщаться так, чтобы его можно было использовать для решения класса задач. Например, программа, написанная для вычисления текущей зарплаты, должна быть написана так, чтобы она могла бы периодически использоваться и в будущем без изменений.

В результате согласования между заказчиком и исполнителем всех перечисленных вопросов составляют техническое задание (ТЗ) в соответствии с ГОСТ 19.201-78, которое служит основанием для дальнейшей работы.

1. **Стадия технического предложения.**

На стадии технического предложения выполняется анализ задачи – это определение и детализация логического порядка действий, которые нужно выполнить над исходными данными, чтобы получить требуемое решение. На этом этапе процесса решения задачи следует в общих чертах описать, что необходимо сделать. Детали того, как это должно быть сделано, будут уточняться на следующем шаге.

*Пример 1*

*Найти корни квадратного уравнения с заданными коэффициентами. Необходимо определить метод решения задачи, будет ли программа находить общее решение или только действительные корни.*

Часто формальная постановка задачи однозначно определяет метод ее решения. Если задача может быть решена несколькими методами, выбирается один из них с учетом сложности реализации, точности результатов и т.д.

1. **Стадия эскизного проектирования.**

На стадии эскизного проектирования при использовании процедурного подхода сложные задачи разбиваются на подзадачи, для которых может строиться своя модель и выбираться свой метод решения. При этом результаты одной подзадачи могут использоваться как исходные данные в другой.

Целесообразно проверить правильность выбранных моделей и методов, выполнив их вручную для некоторых значений исходных данных.

При определении типов исходных данных необходимо также продумать, для каких сочетаний этих данных результат не существует или не может быть получен данным методом, что также надо учитывать при разработке программы.

Одновременно с написанием алгоритма, необходимо точно определить тип и структуру обработанных данных. В одних случаях данными могут быть несколько обычных чисел, в других организация данных будет более сложной.

При определении структуры данных с каждым объектом данных должно быть связано осмысленное имя или идентификатор. При разработке программы идентификаторы будут связаны с расположением данных в памяти.

1. Стадия технического проектирования.
2. Стадия рабочего проектирования.

На стадии рабочего проектирования оформляется программная документация.

Если программу предполагается использовать и сопровождать в течении какого-либо срока, она должна быть документирована. Несмотря на то, что здесь документирование рассматривается как отдельный этап в процессе разработки решения задачи, оно должно выполняться на протяжении всего этого процесса.

Документация должна включать:

* описание постановки задачи;
* описание анализа задачи;
* описание определения структуры данных;
* описание алгоритма;
* текст программы с комментариями;
* тесты с входными и выходными данными;

1. Стадии испытаний и внедрения в производство.
2. Понятие алгоритма. Свойства алгоритма.

Алгоритм — формально описанная последовательность действий‚ которые необходимо выполнить для получения требуемого результата.

* конечность (алгоритм всегда должен заканчиваться после выполнения конечного числа шагов);
* определенность (каждый шаг алгоритма должен быть точно определен);
* ввод (алгоритм имеет некоторое‚ возможно равное нулю‚ число входных данных‚ т.е. величин‚ которые задаются до начала его работы или определяются динамически во время его работы);
* вывод (у алгоритма есть одно или несколько выходных данных‚ т.е. величин‚ которые имеют вполне определенную связь с входными данными);
* эффективность (алгоритм обычно считается эффективным‚ если все его операторы достаточно просты для того‚ чтобы их можно было выполнить в течение конечного промежутка времени с помощью карандаша и бумаги).

1. Основные алгоритмические структуры.
2. Схема и псевдокод алгоритма.

Схемы используются для формального описания алгоритмов. На изображение схем алгоритмов существует ГОСТ 19.701-90, согласно которому каждой группе действий соответствует блок особой формы.

Псевдокод алгоритма — один из способов формального описания алгоритма. Псевдокод базируется на тех же основных структурах, что и структурные алгоритмы. Описать на псевдокоде неструктурный алгоритм нельзя.

Для каждой структуры алгоритма используют свою форму описания.



1. Технология программирования.
2. Структурное программирование.

Структурное программирование рекомендует разбивать (декомпозировать) сложную программу на подпрограммы (процедуры, функции), решающие отдельные подзадачи, т.е. базируется на процедурной декомпозиции.

1. Понятие языка программирования.
2. Синтаксис и семантика языка программирования.

*Синтаксис языка* — совокупность правил, определяющих допустимые конструкции (слова, предложения) языка, его форму.

*Семантика языка* — совокупность правил, определяющих смысл синтаксически корректных конструкций языка, его содержание.

1. Характеристика языка программирования Си.
2. Алфавит языка Си. Идентификаторы.

*Алфавит языка программирования* — множество символов, допустимых при построении конструкций программы.

Алфавит языка Си включает:

* строчные, прописные буквы латинского алфавита (a...z, A...Z) и символ подчеркивания \_ (строчные и прописные буквы различаются)
* цифры (0...9)
* специальные знаки: . , ; + - \* / = < > % & ! ( ) { } ^ | ? : [ ] ^ ~' " # \
* служебные слова: char, short, int, long, float, double, enum, struct, union, signed, unsigned, void, auto, extern, register, static, break, case, continue, default, do, else, for, goto, if, return, switch, while, sizeof, typedef.

Идентификатор — это последовательность букв латинского алфавита (включая символ подчеркивания) и цифр, которая обязательно начинается с буквы. Идентификаторы используются как имена переменных, функций и типов данных. Не рекомендуется начинать идентификатор с символа подчеркивания. Идентификатор не должен совпадать ни с одним ключевым словом языка Си.

1. Основные типы данных языка Си.
2. Константы.

Константа — определенная в программе и неизменяемая величина.

В программе константа может использоваться как поименованная или как литерал.

Поименованная константа – константа, обращение к которой выполняется по имени.

1. Понятия «выражение» и «оператор». Виды операторов в языке Си.
2. Арифметические операции.
3. Преобразования данных при выполнении арифметических операций.
4. Операция присваивания. Составные операции присваивания.
5. Преобразования данных при выполнении операции присваивания.
6. Операция приведения типа.
7. Операции инкрементации и декрементации.
8. Операции отношения.
9. Логические операции.
10. Побитовые операции.
11. Управляющий оператор if-else. Условная операция.
12. Управляющий оператор switch.
13. Управляющие операторы break, continue, goto.
14. Цикл while.
15. Цикл do-while.
16. Цикл for. Операция продолжения ("запятая").
17. Функции. Объявление, определение и вызов функции. Назначение стека.
18. Формальные и фактические параметры функции.
19. Возвращаемое значение функции. Оператор return.
20. Рекурсивные функции.
21. Классы памяти.
22. Внешний класс памяти.
23. Автоматический класс памяти.
24. Статический класс памяти.
25. Регистровый класс памяти.
26. Массивы. Объявление массива. Инициализация массива.
27. Доступ к элементу массива.
28. Массив как аргумент функции.
29. Массив символов.
30. Однотипная обработка одномерных массивов.
31. Переформирование массивов.
32. Одновременная обработка массивов и подмассивов.
33. Поиск в массиве.
34. Сортировка массива. Метод выбора.
35. Сортировка массива. Метод вставки.
36. Сортировка массива. Метод обменов.
37. Указатели: объявление, инициализация, использование.
38. Адресная арифметика.
39. Использование указателя как аргумента функции.
40. Указатель — возвращаемое значение функции.
41. Доступ к аргументам командной строки.
42. Структуры. Объявление структуры. Инициализация структуры.
43. Доступ к элементу структуры (операция "точка").
44. Указатели на структуру. Доступ к элементу структуры (операция "стрелка").
45. Массивы структур. Использование структуры как аргумента функции.
46. Структура - возвращаемое значение функции.
47. Шаги компиляции Си.
48. Препроцессор Си. Поименованные константы.
49. Макросы.
50. Файлы заголовков. Организация сложных программ.
51. Условная компиляция.
52. Библиотеки в ОС UNIX.
53. Буферизованный ввод-вывод.
54. Файл. Открытие и закрытие файла.
55. Файл. Функции чтения-записи.
56. Произвольный доступ к файлу.
57. Виртуальная память пользователя. Функции управления динамической памятью.
58. Динамические структуры данных.