ПОДГОТОВКА К ЭКЗАМЕНУ ПО ООП

C++

1 СТРУКТУРА ПРОГРАММЫ НА ЯЗЫКАХ C и C++. ФУНКЦИИ В C++. ПЕРЕГРУЗКА ФУНКЦИЙ В C++. ПАРАМЕТРЫ ФУНКЦИЙ ПО УМОЛЧАНИЮ.

Структура прог-мы на языке C++ примерно такая же, как и в языке C, за искл. 20-го стандарта. В 20-м стандарте появились модули.

Прог-ма сост. из набора файлов, кот. вкл. в себе объявления и определения.

∃ абстракции ПО:

- данных
  * типы данных - можем объявлять и определять
  * переменные - можем объявлять и определять
  * константы

- действия
  * ф-ции - можем объявлять и определять

Выполнение прог-мы, как правило, начин. с ф-ции с именем „main", т.е. один из файлов реализации должен содержать ф-цию с именем „main". Но по большому счету выполн. прог-мы может начаться с ∀ др. ф-ции и точно также закончиться выполнением ∀ др. ф-ции. Предоставляются прерывания выполн. ф-ций (от 255, из них 0-63 библ. ф-ции)

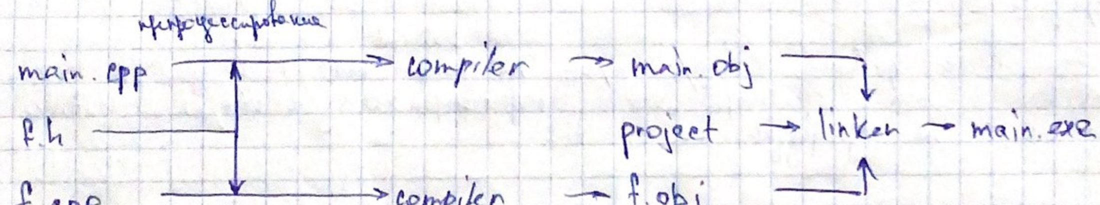
В языке C н/б вложенные ф-ции, т.е. ф-ция может вкл. в себе др. ф-цию. В C++ все ф-ции внешние по отн. друг к другу.

В C/C++ ∀ ф-ция может вызвать ∀ др. ф-цию (в том числе и саму себя). В C++ ф-ция „main" не может вызвать саму себя.

Для исп. файлов реализации, все, что мы хотим использовать, должно сначала определить, либо объявить в том месте, где хотим исп. Для удобства создаются заголовоч. файлы, в кот. выносят объявления.

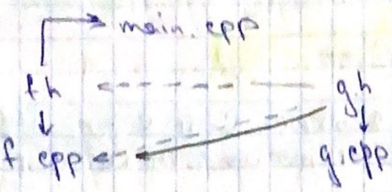
Все файлы, за искл. файла, кот. содерж. ф-цию „main", идут парами.

Процесс сборки:

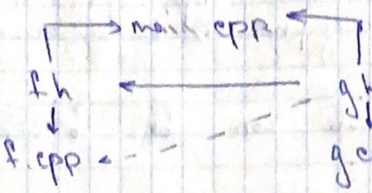


В заголовочном файле категорически нельзя опред. переменные (можно опред.), но можно объявлять, но не надо (глоб. данные); нельзя опред. ф-цию (также как и с перем-ми); можем опред. константы времени компиляции, поскольку на этапе компиляции компилятор подставляет во все места, где встр. константу, знач., и как переменную данную константу нет; можем опред. типы данных, потому, что это высокоуровневая абстракция, разрабатываемая компилятором, т.к.

компилятор приводит это к машинным типам.



подключ. кода к файлам реализ. на этапе разработки прог-много продукта (напр. g.h к f.cpp), чтобы не было лишней перекомпиляции кода.



тут может-е опред-е (ошибка) решение: условное включ-е.

Для защиты от множественного вкл-я заголов. файла исп include guard (директив #ifndef и #define), т.е. каждому заголов. файлу ставится в соотв. с макрос.

Это дает возможность подкл. не только весь, но и часть файла.

Также можно исп. директ. #pragma once (если есть один в всех header-файлах)

Общую часть заголов. файлов выносят в отдельн. заголов. файл. Или не соотв. файлы реализации.

Прог-мма на C/C++ сост. из одной или более подпрограмм.

Ф-ция в языке C/C++ должна иметь свое имя, возвращаемый тип, типы принимаемых параметров и тело ф-ции.

Перегрузка ф-ций — механизм C++, благодаря кот. ф-ции с разным числом или типами параметров могут иметь одинаковые имена.

На перегрузку влияет:

- разное число прним. параметров;
- разн. тип переменных и так наз. интерпретации в том
- const значение ф-ции/метода. Если объект константый, то будет вызов. const метод, иначе не const.

Тип возвращ. переменной не имеет значения.

В C++ один или несколько аргументов ф-ции могут задаваться по умолчанию. Должны быть последними в списке параметров. Инициализируются на этапе компиляции.

Если есть объявление ф-ции, то знач. параметров по умолчанию устанав. в объявлении. При определении этого делать не будем.

Если нет объявления ф-ции, то знач. параметров по умолч. установ. при определении.

Если есть ф-ция с параметром по умолчанию и ф-ция с таким же кол-вом и типом параметров, но без параметра по умолчанию, то это неоднозначность, а не перегрузка.