Разделение текста программы на модули

Разделение исходного текста программы на несколько файлов становится необходимым по многим причинам:

1. С большим текстом просто неудобно работать.
2. Разделение программы на отдельные модули, которые решают конкретные подзадачи.
3. Разделение программы на отдельные модули, с целью повторного использования этих модулей в других программах.
4. Разделение интерфейса и реализации.

Интерфейс и реализация

Когда часть программы выделяется в модуль (единицу компиляции), то компилятор должен знать, что имеется в этом модуле. Для этого служат заголовочные файлы.

Таким образом, модуль состоит из двух файлов: заголовочного (интерфейс) и файла реализации.

Заголовочный файл, как правило, имеет расширение `.h` или `.hpp`, а файл реализации — `.cpp` для программ на C++ и `.c`, для программ на языке C (в STL включаемые файлы без расширений, но, они являются заголовочными файлами).

Заголовочный файл должен содержать все объявления, которые должны быть видны снаружи. Объявления, которые не должны быть видны за пределами заголовочного файла, делаются в файле реализации.

Что может быть в заголовочном файле

Правило 1. Заголовочный файл может содержать только объявления. Заголовочный файл не должен содержать определения.

При обработке содержимого заголовочного файла компилятор ***не должен*** генерировать информацию для объектного модуля.

Правило 2. Заголовочный файл должен иметь механизм защиты от повторного включения.

2.1. Для этих целей может применяться директива `#pragma once`.

2.2. Защита от повторного включения реализуется директивами препроцессора (преимущество второго способа в том, что он работает на любых компиляторах):

```
#ifndef SYMBOL
#define SYMBOL

// набор объявлений

#endif
```

Пояснение. При первом включении заголовочного файла в исходный код для препроцессора условие **`#ifndef SYMBOL`** (означает: "символ `SYMBOL` не определён") истинно. Следующая директива препроцессора определяет символ `SYMBOL` – **`#define SYMBOL`** и обрабатывает все строки до директивы препроцессора **`#endif`**. Повторного включении не произойдет, поскольку условие `#ifndef SYMBOL (`" символ `SYMBOL` не определён") ложно (т.к. символ был определён непосредственно при первом включении), и весь набор объявлений пропускается. Закрывающая директива – **`#endif`**.

В качестве **`SYMBOL`** обычно применяют имя самого заголовочного файла в верхнем регистре, обрамлённое одинарными или сдвоенными подчерками. Например, для файла ***header.h*** традиционно используется **`#define __HEADER_H__`**.

Что может быть в файле реализации

Файл реализации может содержать как определения, так и объявления. Объявления, находящиеся в файле реализации, являются лексически локальными для этого файла. Т.е. будут действовать только для этой единицы компиляции.

Правило 3. В файле реализации должна быть директива включения соответствующего заголовочного файла.

Понятно, что объявления, которые видны снаружи модуля, должны быть также доступны и внутри.

Правило также гарантирует соответствие между описанием и реализацией. При несовпадении, например, сигнатуры функции в объявлении и определении компилятор выдаст ошибку.

Правило 4. В файле реализации не должно быть объявлений, дублирующих объявления в соответствующем заголовочном файле.

При выполнении **Правила 3**, нарушение **Правила 4** приведёт к ошибкам компиляции.