Модульное программирование

Мо́дульное программи́рование — это организация программы как совокупности небольших независимых блоков, называемых модулями, структура и поведение которых подчиняются определенным правилам.[1] 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.

Мо́дуль — функционально законченный фрагмент программы. Во многих языках (но далеко не обязательно) оформляется в виде отдельного файла с исходным кодом или поименованной непрерывной её части. Некоторые языки предусматривают объединение модулей в пакеты.

1. Модульность программного кода

Принцип модульности является средством упрощения задачи проектирования ПС и распределения процесса разработки ПС между группами разработчиков. При разбиении ПС на модули для каждого модуля указывается реализуемая им функциональность, а также связи с другими модулями.[2]. Удобство использования модульной архитектуры заключается в возможности обновления (замены) модуля, без необходимости изменения остальной системы.

Роль модулей могут играть структуры данных, библиотеки функций, классы, сервисы и др. программные единицы, реализующие некоторую функциональность и предоставляющие интерфейс к ней.

Программный код часто разбивается на несколько файлов, каждый из которых компилируется отдельно от остальных. Такая модульность программного кода позволяет значительно уменьшить время перекомпиляции при изменениях, вносимых лишь в небольшое количество исходных файлов, и упрощает групповую разработку. Также это возможность замены отдельных компонентов (таких как jar-файлы, so или dll библиотеки) конечного программного продукта, без необходимости пересборки всего проекта (например, разработка плагинов к уже готовой программе).

Одним из методов написания модульных программ является объектно-ориентированное программирование. ООП обеспечивает высокую степень модульности благодаря таким свойствам, как инкапсуляция, полиморфизм и позднее связывание.

2. Модульная система модулей

Несмотря на то, что модульное программирование никак не связано с деталями конкретного языка (и даже в случае отсутствия явной поддержки со стороны языка может применяться при достаточной дисциплине со стороны программистов), большинство языков выдвигают на верхний уровень свою собственную природу системы модулей, словно перенос системы модулей с одного языка на другой был бы невозможен[3].

В 2000 году Ксавье Лерой предложил делать системы модулей модульными, то есть параметризуемыми описанием конкретного ядра языка со своей системой типов[3]. В качестве примера он продемонстрировал обобщённую реализацию языка модулей ML (как наиболее развитой системы модулей из известных на данный момент) и примеры её инстанцирования на традиционный для неё язык ML и на язык Си.

Реализация Лероя сама построена посредством языка модулей ML, а именно в виде функтора, параметризованного данными о ядре языка и описанием его механизма проверки согласования типов. Это значит, что при написании компилятора некоторого языка достаточно описать ядро языка и передать его данному функтору (как библиотечной функции) — в результате получится компилятор расширения известного языка системой модулей ML.

3. История концепции модулей

История концепции модулей как единиц компиляции восходит к языкам Фортран II и Кобол, то есть, к концу 1950-х годов[4][5]. В 1976 году появилась публикация, в которой была развита концепция модульности — о языке Mesa (*англ.*), который был разработан в Xerox PARC. В 1977 году подробно ознакомился с этой концепцией ученый Никлаус Вирт, общаясь с разработчиками в Xerox PARC.[6] Эти идеи были использованы Виртом при создании языка Модула−2, публикация о котором вышла в 1977 году[7].

Термин «модуль» в программировании начал ис-

пользоваться в связи с внедрением модульных принципов при создании программ. В 1970-х годах под модулем понимали какую-либо процедуру или функцию, написанную в соответствии с определенными правилами. Например: «модуль должен быть простым, замкнутым (независимым), обозримым (от 50 до 100 строк), реализующим только одну функцию задачи, имеющим одну входную и одну выходную точку».

Первым основные свойства программного модуля более-менее четко сформулировал Д. Парнас (David Parnas) в 1972 году: «Для написания одного модуля должно быть достаточно минимальных знаний о тексте другого». Таким образом, в соответствии с определением, модулем могла быть любая отдельная процедура (функция) как самого нижнего уровня иерархии (уровня реализации), так и самого верхнего уровня, на котором происходят только вызовы других процедур-модулей.[8]

Таким образом, Парнас первым выдвинул концепцию скрытия информации (англ. *information hiding*) в программировании. Однако существовавшие в языках 70-х годов только такие синтаксические конструкции, как процедура и функция, не могли обеспечить надежного скрытия информации, из-за повсеместного применения глобальных переменных.

Решить эту проблему можно было только разработав новую синтаксическую конструкцию, которая не подвержена влиянию глобальных переменных. Такая конструкция была создана и названа модулем. Изначально предполагалось, что при реализации сложных программных комплексов модуль должен использоваться наравне с процедурами и функциями как конструкция, объединяющая и надежно скрывающая детали реализации определенной подзадачи.

Таким образом, количество модулей в комплексе должно определяться декомпозицией поставленной задачи на независимые подзадачи. В предельном случае модуль может использоваться даже для заключения в него всего лишь одной процедуры, если необходимо, чтобы выполняемое ею локальное действие было гарантировано независимым от влияния других частей программы при любых изменениях.

Впервые специализированная синтаксическая конструкция модуля была предложена Н. Виртом в 1975 г. и включена в его новый язык Modula . Насколько сильно изменяются свойства языка, при введении механизма модулей, свидетельствует следующее замечание Н.Вирта, сделанное им по поводу более позднего языка Модула−2: «Модули — самая важная черта, отличающая язык Модула−2 от его предшественника Паскаля».

4. Реализация в языках программирования

Языки, формально поддерживающие концепцию модулей: IBM S/360 Assembler, Кобол, RPG, ПЛ/1, Ада, D, F (англ.), Фортран, Haskell, Blitz BASIC, OCaml, Паскаль, ML, Модула−2, Оберон, Компонентный Паскаль, Zonnon, Erlang, Perl, Python и Ruby. В IBM System использовались «модули» от языков RPG, Кобол и CL, когда программировалась в среде ILE.

Модульное программирование может быть осуществлено, даже когда синтаксис языка программирования не поддерживает явное задание имён модулям.

Программные инструменты могут создавать модули исходного кода, представленные как части групп — компонентов библиотек, которые составляются с программой компоновщиком.

Стандартный Паскаль не предусматривает механизмов раздельной компиляции частей программы с последующей их сборкой перед выполнением. Вполне понятно стремление разработчиков коммерческих компиляторов Паскаля включать в язык средства, повышающие его модульность.[9]

Модуль в Паскале — это автономно компилируемая программная единица, включающая в себя различные компоненты раздела описаний (типы, константы, переменные, процедуры и функции) и, возможно, некоторые исполняемые операторы инициирующей части.[10]

По своей организации и характеру использования в программе модули Паскаля близки к модулям-пакетам (PACKAGE) языка программирования Ада. В них так же, как и в пакетах Ады, явным образом выделяется некоторая «видимая» интерфейсная часть, в которой сконцентрированы описания глобальных типов, констант, переменных, а также приводятся заголовки процедур и функций. Появление объектов в интерфейсной части делает их доступными для других модулей и основной программы. Тела процедур и функций располагаются в исполняемой части модуля, которая может быть скрыта от пользователя.

Модули представляют собой прекрасный инструмент для разработки библиотек прикладных программ и мощное средство модульного программирования. Важная особенность модулей заключается в том, что компилятор размещает их программный код в отдельном сегменте памяти. Длина сегмента не может превышать 64 Кбайт, однако количество одновременно используемых модулей ограничивается лишь доступной памятью, что позволяет создавать большие программы.

5. Примечания

[1] http://vit-prog.narod.ru/page/TRPP/section_1/subject_1.3.htm

[2] МОДУЛЬНОЕ ПРОГРАММИРОВАНИЕ - Визуальный словарь. Проверено 18 апреля 2013. Архивировано из первоисточника 19 апреля 2013.

[3] Leroy - A Modular Module System

[4] A brief history of FORTRAN

[5] COBOL Subprograms

[6] Никлаус Вирт. Краткая история Modula и Lilith, перевод с англ. с комментариями в тексте Р. Богатырева

[7] The History of Modula-2 and Oberon

[8] *D. L. Parnas* On the criteria to be used in decomposing systems into modules (англ.) // Communications of the ACM. — 1972. — Vol. 15, no. 12. — DOI:10.1145/361598.361623.

[9] http://www.pascal.helpov.net/index/pascal_modules_programming [*неавторитетный источник? 856 дней*]

[10] Павловская Татьяна Александровна. Язык программирования Pascal (учебный курс)

6. См. также

- Интерфейс программирования приложений
- Абстрактный тип данных
- Язык модулей ML

7. Литература

- *Xavier Leroy.* A Modular Module System // vol.10, issue 3. — Journal of Functional Programming, 2000. — С. 269-303.

8. Источники текстов и изображения, авторы и лицензии

8.1. Текст

- **Модульное программирование** *Источник:* https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5?oldid=76740394 *Авторы:* Vladimir Solovjev, Don Reba, Knyf, РоманСузи, Atr2006, Lit-uriy, Иветта, Arachnelis, РобоСтася, InDigazzZ, AVB, Vadimr, Alex KMB, Blackb00m, Rufus Total, Tretyak, Dmitry Bagroff, ESSch, Onliner~ruwiki, AVL 93, MerlIwBot, W2Bot, Rasulzhan, Ai jedi, Addbot, Oleg3280, Эсто Грано, Dima Borisenko, Glovacki, YiFeiBot, Oguretsi и Аноним: 10

8.2. Изображения

8.3. Лицензия