Структура программы. Стиль программирования.

В программе программист записывает последовательность действий, выполняемых над определенными данными с помощью различных операций для реализации заданной цели. Основные требования, предъявляемые к программе:

* точность полученного результата;
* время выполнения;
* объем требуемой памяти.

Максимальный размер программы ограничен. [Компилятор](http://moodle32.lms.tpu.ru/mod/glossary/showentry.php?eid=1528&displayformat=dictionary) позволяет обрабатывать программы, в которых объем данных и генерируемый машинный код не превышают 64 Кбайт каждый. Если объем программы требует большего количества памяти, то необходимо использовать дополнительные средства.

Структура программы:

1. Заголовок, состоящий из зарезервированного слова **program** и имени программы. Заголовок несет смысловую нагрузку и может отсутствовать, однако рекомендуется всегда его записывать для быстрого распознавания нужной программы.
2. Раздел описаний, в котором должны быть описаны все идентификаторы, встречающиеся в программе. Он представляет собой:

* список имен подключаемых библиотечных модулей (определяется зарезервированным словом **uses**);
* описание меток (**label**);
* описание констант (**const**);
* определение типов данных (**type**);
* описание переменных (**var**);
* описание процедур и функций.

Раздел описания начинается соответствующим каждому элементу списка служебным словом (табл. 1), после которого идет последовательность величин одного типа, разделенных запятой. После списка имен ставится двоеточие и указывается тип данных (см. 1.2).

Любой элемент данного списка в программе может отсутствовать.

Программа состоит из заголовка и блока.

## **Заголовок программы**

В заголовке указывается имя программы и список параметров. Общий вид:

**program** n (input, output, x, y,…);

здесь n – имя программы; input – файл ввода; output – файл вывода; x, y – внешние файлы, используемые в программе.

Заголовка может и не быть или он может быть без параметров.

## **Блок**

Блок программы состоит из шести разделов, следующих в строго определенном порядке:

1. раздел меток (label)
2. раздел констант (const)
3. раздел типов (type)
4. раздел переменных (var)
5. раздел процедур и функций
6. раздел действий (операторов).

Раздел действий должен присутствовать всегда, остальные разделы могут отсутствовать.

Каждый из первых четырех разделов начинается с соответствующего ключевого слова (label, const, type, var), которое записывается один раз в начале раздела и отделяется от последующей информации только пробелом, либо концом строки, либо комментарием.

### **Раздел меток (label)**

Любой выполняемый оператор может быть снабжен меткой – целой положительной константой, содержащей не более 4-х цифр. Все метки, встречающиеся в программе, должны быть описаны в разделе label.

Общий вид:

**label** l1, l2, l3…;

здесь l1, l2, l3 – метки.

**Пример.** label 5, 10, 100;

Метка отделяется от оператора двоеточием.

Пример. Пусть выражение a := b имеет метку 20. Тогда этот оператор выглядит так:

20: a := b;

### **Раздел констант (const)**

Если в программе используются константы, имеющие достаточно громоздкую запись (например, число пи с 8-ю знаками), либо сменные константы (для задания варианта программы), то такие константы обычно обозначаются какими-либо именами и описываются в разделе const, а в программе используются только имена констант. Это делает программу более наглядной и удобной при отладке и внесении изменений.

Общий вид:

**const** a1 = c1; a2 = c2; …

здесь a1 – имя константы, c1 – значение константы.

Пример. const pi = 3.14; c = 2.7531;

### **Раздел типов (type)**

Если в программе вводится тип, отличный от стандартного, то этот тип описывается в разделе type:

**type** t1 = вид\_типа;

t2 = вид\_типа;

где t1 и t2 – идентификаторы вводимых типов.

Затем тип используется при объявлении переменных.

Пример использования нестандартных типов:

**const**

len=40;

**type**

year=1930..2010;

names=**string**[len];

**var**

empl: names;

y: year;

Раздел описания типов имеет большое значение в программе на языке Pascal. Если в программе не использовать типы, то можно столкнуться с несовместимостью типов переменных, даже если они описаны одинаково.

### **Раздел переменных (var)**

Пусть в программе встречаются переменные v11, v12,…; все они должны быть описаны следующим образом:

**var** v11, v12,…: type1;

v21, v22,…: type2; …

здесь v11, v12,… - имена переменных; type1 – тип переменных v11, v12,…; type2 – тип переменных v21, v22,….

Пример. var k, i, j: integer; a, b: real;

Каждая переменная должна быть описана до ее использования в программе и отнесена к одному и только одному типу. Названия разделов (const, type, var…) указываются только один раз.

Пример.

**var** a: **real**;

b: **real**;

Таким образом, в разделе var вводится имя каждой переменной и указывается, к какому типу эта переменная принадлежит. Тип переменной можно задать двумя способами: указать имя типа (например, real, color и т.д.), либо описать сам тип, например: array[1..16] of char

### **Раздел процедур и функций**

Здесь присутствуют заголовки и тела пользовательских процедур и функций.

### **Раздел действий (операторов)**

Эта часть программы начинается с ключевого слова begin и заканчивается словом end, после которого должна стоять точка. Раздел действий есть выполняемая часть программы, состоящая из операторов.

Рассмотрим примеры

Любая программа на языке программирования Pascal имеет три составляющие: 1) заголовок; 2) раздел описаний; 3) тело программы.

✎ 1) ***Заголовок*** – это слово **Program** , после которого идет название программы латиницей и точка с запятой в конце. Название не может начинаться с цифры, не должно совпадать с зарезервированными словами (begin, end, integer и т.п.) или с названиями переменных из раздела описаний (см. ниже), а также недопустимо использование каких бы то ни было символов (‘@’, ‘%’, ‘&’ и т.п.), кроме подчеркивания. Название желательно давать со смыслом, которое отображало бы суть самой программы. Но заголовок писать не обязательно.

✎ 2) ***Раздел описаний*** – Поскольку сначала мы будем рассматривать простейшие задачи, то у нас во втором разделе будет находиться или описание констант, или переменных.

✎ 3) ***Тело программы*** – блок операторов, в котором записываются команды для выполнения. Начинается этот блок словом **begin** («начало»), а заканчивается словом **end.** («конец») с точкой в конце.

Таким образом, в общем случае программа на Паскале выглядит следующим образом:

**Код Pascal**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10 | **Program** <название программы>;  <раздел описаний>  **begin**  <оператор 1>;  <оператор 2>;  . . . . . . .  <оператор N>;  **end**. | |

При решении простейших задач ограничимся сначала целыми числами. Перед тем как начать программировать, укажем простейшие 4 действия над числами в Паскале: «+» − сложение, «-» − вычитание, «\*» − умножение, «/» − деление. Как видим, в программировании и математике элементарные действия над числами обозначаются одинаково. Решим следующую задачу:

**Задача.** Пусть даны целые числа A и B. Найти их сумму.

Назовем программу нахождения суммы как Summa. Тогда часть кода, отвечающая за название программы, будет иметь вид:

**Код Pascal**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1 | **Program** Summa; *{ программа называется Summa }* | |

Обозначим сумму чисел A и B через S. Поскольку числа целые, то и сумма будет целым числом. Таким образом, в описании чисел A, B, S будем использовать переменную целого типа integer. Но как описать эти числа в разделе описаний – как константы (**const**) или как переменные (**var**)? Все зависит от условия задачи. Мы можем числа A и B записать и как константы, и как переменные. Но в любом случае сумма S будет переменного типа, поскольку нам неизвестно её значение.

Итак, пусть дано значение чисел A и B, равное, например, 23 и 76 соответственно. Поскольку мы знаем значения наперед, то A и B будут типа **const**, а S опишем как **var**. Учитывая, что описание констант идет перед описанием переменных, раздел описаний в этом случае будет следующим:

**Код Pascal**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6 | **Const**  A = 23;  B = 76;  **var**  S: **integer**; | |

Таким образом, тело программы будет состоять из вычисления суммы S и вывода её на экран. В паскале, чтобы вычислить сумму А и В, нельзя просто записать S = A + B; такая запись возможна, если надо проверить, не равно ли число S сумме A + B. А ***чтобы в переменную S записать значение суммы А + В, надо значение этой суммы «присвоить» переменной S***, т.е. сумму А + В записать в то место памяти компьютера, которое выделено для переменной S. Это как бы «впихивание» значения одного числа в значение другого. Записывается [присваивание](https://progmatem.ru/pascal/operator-prisvaivaniya.html) так:

* S := A + B;

Для вывода результата используется оператор вывода write с комментарием на экране:

write('Сумма чисел 23 и 76 равна: S = ', S);

Объединяя наши 1)заголовок программы, 2)раздел описаний и 3)тело программы, получим небольшой код, который можно скопировать в PascalABC.Net или в другую версию Pascal:

**Код Pascal**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | **Program** Summa; *{ программа называется Summa }*  **Const**  A = 23;  B = 76;  **var**  S: **integer**;  **begin**  S := A + B; *{ сумма чисел A и B }*  writeln('Сумма чисел 23 и 76 равна: S = ', S);  readln *{ <-- Ожидание нажатия Enter, после*  *которого программа завершится }*  **end**. | |

После выполнения этой программы на экране мы увидим:

Сумма чисел 23 и 76 равна: S = 99

Все хорошо и программа работает. Но здесь мы написали кое-что лишнее. Посмотрите внимательно на оператор вывода write в конце – там выводится сумма S. Изменится ли что-нибудь, если вместо S написать само значение суммы, то есть А + В? Правильно, ничего. Только тогда надо убрать переменную S из нашего кода, а вместо неё при выводе на экран использовать сумму А + В. Вот упрощенный код:

**Код Pascal**

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | 1  2  3  4  5  6  7  8  9  10 | **Program** Summa; *{ программа называется Summa }*  **Const**  A = 23;  B = 76;  **begin**  writeln('Сумма чисел 23 и 76 равна', A + B); *{ <-- Выводим сумму }*  readln *{ <-- нажимаем Enter и выходим из программы }*  **end**. | |

В этом случае результат на экране будет такой:

Сумма чисел 23 и 76 равна 99

Это и есть наша простейшая первая программа.

Стиль программирования.

Под ***стилем программирования*** понимается набор приемов или методов, направленных на создание правильных, эффективных, удобных для использования программ. Другими словами, это ключ к красиво написанным, хорошо отлаженным программам. Ведь если программист строго придерживается какого-либо стиля, значит и "видит" свою программу лучше и ошибок делает меньше, и сопровождение этой программы не превращается в кошмар. Если не заниматься выработкой стиля программирования с самого начала, то постепенно разгильдяйское отношение к своему исходному тексту войдет в привычку, избавиться от которой впоследствии будет очень сложно.

**Программа***–* это документ для последующего использования, корректировки, модификации. Поэтому языки программирования должны обеспечивать возможность создания удобочитаемых предложений программы. Программисты независимо от времени всегда должны понимать не только свои, но и чужие программы, и в этом им должны помогать элементы стиля.

**Элементами стиля являются**: комментарии, пропуск строк, использование пробелов, выбор имен переменных и файлов, стандартные сокращения, перенос имен или литералов на другую строку, использование скобок, отступов от начала строки и др. Рассмотрим их более подробно.

**Программы с пояснительными комментариями** значительно легче отлаживать, так как они содержат дополнительную информацию для работы с программой. Не комментируемая программа – это, вероятно, наихудшая ошибка, которую может сделать программист, а также свидетельство его дилетантского подхода. Комментарии подобны ориентирам в незнакомом лесу.

Хорошее правило – включать комментарии в процессе написания программы. Однако хорошие комментарии написать не просто, так как они должны быть также хорошо продуманы и проработаны, как и кодировка программы. Комментарии нужны как на стадиях проектирования и отладки программы, так и позже. Существует три типа комментариев: вводные, оглавления и пояснительные.

В***вводные комментарии*** обычно помещают информацию о назначении программы, указания по ее вызову и использованию, о ее структуре, о требуемом объеме памяти, о времени выполнения программы, дате написания программы, сведения об авторе и др.

**Оглавление** в виде комментариев делается в начале больших программ. Оглавление должно содержать название, размещение и функцию каждого программного модуля, снабженного именами и комментариями, указывающими его назначение.

**Пояснительные** комментарии должны сопровождать трудно понятные части программы. Перед циклами или условными операторами, важными для понимания логики программы, должны появляться комментарии с указанием действия, которое будет производиться. Средней нормой можно считать одну строку комментариев на десять строк программы, написанной на языке высокого уровня. Комментарии должны объяснять цель группы операторов, а не описывать действия, производимые этими операторами. Комментарии должны содержать некоторую дополнительную информацию, а не перефразировать программу.

Комментарии должны быть правильными с самого начала и изменяться в соответствии с изменениями программы. Очевидно, что *неправильные комментарии* – это хуже, чем их отсутствие, поскольку они вводят в заблуждение программиста.

**Пропуск строк** – это часто недостаточно оцениваемый метод улучшения наглядности программ. Этот метод можно использовать для вертикальной разрядки. Пропуском одной строки можно отделять каждую группу логически связанных операторов, пропуском двух строк – основные логические фрагменты программы.

**Пробелы** следует ставить везде, где это приводит к улучшению читаемости программы.

**Выбор имен переменных** выполняется так, чтобы наилучшим образом отобразить те величины, которые они представляют. Если ограничения на размер имени отсутствуют, то целесообразно использовать настолько длинные имена, насколько это нужно. Правильный выбор имен переменных – это залог удобочитаемости программ, так как он требует незначительных умственных усилий от программиста и столь же небольшого расхода машинного времени. Рекомендуется в качестве имен переменных использовать термины предметной области. В следующем фрагменте при определении площади прямоугольника используются несуразные идентификаторы, не имеющие ничего общего с предметной областью: