Олександр Павленко
аспірант ТДАТУ, старший викладач кафедри системного аналізу
Мелітопольський інститут державного та муніципального управління
«Класичного приватного університету» м. Мелітополь

Побудова великих проектів та основні властивості підпрограм в середовищі Object Pascal

Оригінальний текст великих проектів на Object Pascal складається зі спеціально оформлених блоків, які називаються підпрограмами. Підпрограми реалізують окремі фрагменти алгоритму розв'язання великої задачі.

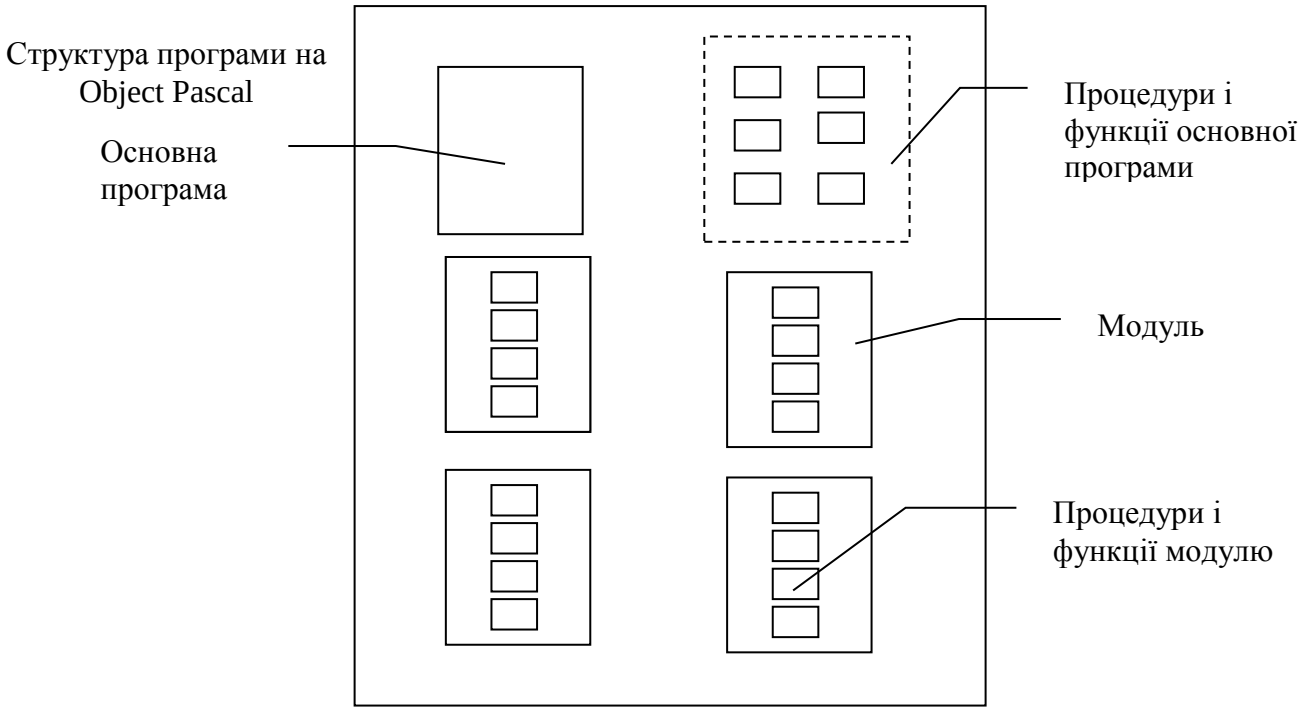
Застосування підпрограм дозволяє:

1. Створити впорядковану структуру програми, що поліпшує розробку програми, пошук помилок, модифікацію і т.д.
2. Скоротити вихідний текст програми. Повторювані ділянки програми оформляються у вигляді підпрограм. Потім самі ділянки тексту замінюються на виклики підпрограм.

Для виклику підпрограми в тексті програми згадується ім'я підпрограми. При виклику підпрограми виконання поточної програми переривається. Потім виконуються оператори підпрограми. Після закінчення виконання підпрограми починає виконуватися програма з перерваного місця.

Підпрограма повинна мати наступні властивості:

1. Функціональність. Підпрограма має виконувати чітко визначену функцію і не робити нічого стороннього.
2. Самостійність («ізольованість»). Підпрограма має якомога менше залежати від зовнішніх даних, а виконувати дії на основі вхідних параметрів та внутрішніх даних. Результати роботи оформлюються у вигляді вихідних параметрів і не зачіпають якихось зовнішніх глобальних даних.

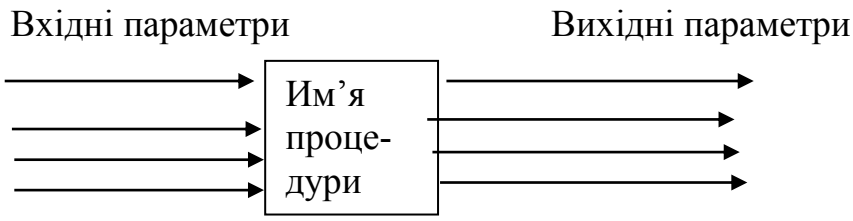


Підпрограми Object Pascal бувають двох видів: функції і процедури.

1. Функція.

Функція повертає один результат, що дозволяє використовувати її у висловах.

2. Процедура.



Процедура - сукупність операторів виконують деякі дії над вхідними параметрами і повертає результат у вигляді вихідних параметрів в місце виклику.

Розглянемо випадок процедур та функцій основної програми. Перед використанням процедури і функції описуються в блоці описів основною програмою.

Формат опису:

Опис функції починається з заголовка:

Function ім'я функції (список формальних параметрів): тип; тіло функції; ім'я функції - те ім'я, яке програміст вирішив дати своєї функції.

Список формальних параметрів - список змінних із зазначенням їх типів, які використовуються для передачі функції вхідних даних.

Тип - тип повертається результату (тип значення яке обчислює функція). Це прості типи, покажчики і рядки.

Тіло функції - за структурою нічим не відрізняється від програми (крім розділу uses), але закінчується не крапкою, а крапкою з комою.

Приклад:

Опис і використання функції вибору більшого з двох чисел.

```
/ / Y = (a + b) 2 + a
function fy (a, b: real): real;
begin
fy: = sqr (a + b) 2 + a;
end;
/ / Основна програма
begin
writeln ( 'введіть 2 числа X і Y');
readln (x, y);
z: = fy (x, y);
writeln ( 'Більшу з 2 чисел', z);
readln;
end.
```

Приклади заголовків:

```
Function myf1 (a: real; b: real; c: boolean): boolean;
Function myf2 (a, b: real; c, d: integer): byte;
```

Формальні параметри в списку розділяються крапкою з комою. Якщо параметри одного типу, то допускається перерахування через кому з зазначенням одного слова типу.

```
Function myfun (a: real; b: real; c: real): real;
```

Function myfun (a, b, c: real): real;

Повернення результату. Функція повертає єдиний результат в місце виклику через своє ім'я. У зв'язку з цим всередині функції повинен бути присутнім наступний оператор:

ім'я функції: = вираз;

або

result: = вираз;

Змінна *result* вважається описаною.

Виклик функції. Після опису функції її можна використовувати у виразах поряд зі стандартними функціями.

Формат виклику: ім’я функції (список фактичних параметрів). Список фактичних параметрів – список аргументів функції, що представляє собою вираження, порядок проходження і тип яких співпадає з формальними параметрами в заголовку функції.

Приклад

```
function myfunction (a, b: real; c: integer; d: boolean): real;
var ...
begin
...
result: = ...;
end;
var x, y: real;
z: integer;
b: boolean;
begin
a: = 7 5 * myfunction (x, y +0.7, z-3, b)
end.
```

При виконанні функції створюються формальні змінні і їм присвоюються в якості початкових значень значення фактичних параметрів:

a: = x, b: = y +0.7, c: = z-3, d: = b

При виконанні функції виконуються наступні дії:

1. обчислюються значення виразів фактичних параметрів;
2. значення виразів присвоюються створеним формальним параметрам;
3. виконується тіло функції та оператор повернення результату;
4. результат функції ставиться у вихідне вираз на місце звернення до функції;
5. обчислення початкового виразу триває.

Процедури. Використовуються в тому випадку, коли підпрограма повертає багато значень результату або не повертає нічого.

Формат опису:

procedure ім’я процедури (список формальних параметрів); тіло процедури

У заголовку відсутня вказівка типу результату. Процедури повертають результати по іншому. Список формальних параметрів процедури складається як з вхідних параметрів, так із вихідних. Все інше в описі процедури теж.

Приклад

```
procedure sum (a, b: integer);
begin
writeln ( 'Сума a і b дорівнює', a + b)
end;
```

Виклик процедури

Звернення до процедури відбувається за допомогою оператора виклику процедури:

ім’я процедури (список фактичних параметрів);

Список фактичних параметрів повинен відповідати списку формальних в заголовку процедури.

Параметри процедур та функцій

Параметри бувають:

1. формальні - знаходяться в заголовку процедур і функцій;
2. фактичні - вказуються при виклику процедур і функцій.

Формальні параметри:

1. параметри-значення;
2. параметри-змінні;
3. параметри-константи.

Параметри-значення - використовуються як вхідні для підпрограм. Параметри-змінні використовуються в процедурах для повернення значень результату. Параметри-константи - використовуються як вхідні для процедур і функцій. Для того, щоб повернути значення з процедури потрібно вказати в заголовку один або декілька параметрів змінних:

```
procedure sum (a, b: integer; var c: integer);
begin
c: = a + b
end;
begin
...
sum (x, y, z); / / z: = x + y
writeln (z);
...
end.
```

Література

1. Довгаль С.И., Литвинов Б.Ю., Сбитнев А.И. Персональные ЭВМ: Турбо-Паскаль V7.0, Объектное программирование, Локальные сети. (Учебное пособие).- Киев, «Информсистема сервис», 1993.

2. Епанешников А.М., Епанешников В.А. Программирование в среде Turbo-Pascal 7.0 .- М.:, Диалог МИФИ, 1993.

3. Поляков Д.Б., Круглов Н.Ю. Программирование в среде Турбо-Паскаля. - изд.МАИ., М.:, 1992.

4. Фаронов В.В. Турбо-Паскаль. Начальный курс – 1 кн. Практика программирования –2 кн. Учебное пособие. - М.: «Нолидж»,1997.

5. Верещага В. М., Конопацький Є. В., Павленко О. М. Визначення площі, обмеженої топографічною замкненою плоскою кривою //Комп'ютерно-інтегровані технології: освіта, наука, виробництво. – 2015. – №. 20. – С. 119-123.

6. Павленко О. М. Геометричне моделювання вертикального планування горизонтальної земельної ділянки засобами точкового БН-числення: дис. – Мелітопольський державний педагогічний університет імені Богдана Хмельницького, 2017.

7. Верещага, В. М. Спосіб згортання (розгортання) чарунок [Текст] / В. М. Верещага, Є. О. Адоньєв, О. М. Павленко // Сучасні проблеми моделювання. – 2016. – №. 7. – С. 32–38.

8. Павленко О.М., Верещага В.М., Кучеренко В.В.Вертикальне планування на місцевості земельної ділянки до ідеально горизонтальної площини // Сучасні проблеми моделювання. – 2014. – №. 5. – С. 32–38.