***Поняття основного та допоміжного алгоритму. Алгоритми-процедури і алгоритми-функції. Формальні та фактичні параметри***

**Конспект уроку. 11-А клас. 12.02.13р.**

**Провів Бондарчук Костянтин**

***Тема уроку:*** **"Поняття основного та допоміжного алгоритму. Алгоритми-процедури і алгоритми-функції. Формальні та фактичні параметри"**

***Мета уроку:*** Дати поняття про допоміжні алгоритми, типи допоміжний алгоритмів, оформлення підпрограм (процедур) мовою програмування Паскаль, поняття про формальні та фактичні параметри, області дії змінних.
***Тип уроку:***Лекційний.

***Теоретичний матеріал:***
**Підпрограмою** називається іменована, логічно закінчена група операторів мови, яку можна викликати для виконання будь-яку кількість разів із різних місць програми.
У мові Паскаль для організації підпрограм використовуються процедури і функції.
**Процедура** - це незалежна поіменована частина програми, призначена для виконання визначених дій. Вона складається з тіла і заголовка. За структурою її можна розглядати як програму в мініатюрі. Після однократного опису процедуру дозволяється викликати за іменем з наступних частин програми. Використання імені процедури в програмі називається викликом процедури. Ім'я процедури не може знаходитися у виразі у якості операнду.
**Функція** відрізняється від процедури тим, що, по-перше, передає в точку виклику скалярне значення (результат своєї роботи), а по-друге, ім'я функції може входити у вирази, як операнд. Функція, якщо вона зустрічається у виразі, називається покажчиком функції або звертанням до функції.
Усі процедури і функції мови Паскаль підрозділяються на дві групи:
вбудовані;
визначені користувачем.
**Вбудовані (стандартні) процедури** і функції є частиною мови і можуть викликатися за іменем без попереднього опису в розділі описового блока. З багатьма з них ви вже знайомилися в попередніх розділах.
**Процедури і функції користувача** організовуються самим програмістом відповідно до синтаксису мови і являють собою локальні блоки. Попередній опис процедур і функцій користувача є обов'язковим.
**Функція**, визначена користувачем, складається з заголовка і тіла функції. Заголовок містить зарезервоване слово **Function**, ідентифікатор (ім'я) функції та, укладений у круглі дужки, необов'язковий список формальних параметрів і тип значення, що повертається функцією.
**Формат опису:**
*Function <ім'я> [(формальні параметри)]:<тип результату>;*
***Ім'я функції*** - унікальний у межах блока ідентифікатор. Результат, що повертається, може мати будь-який простий тип і тип *string.*
Тіло функції являє собою локальний блок, за структурою аналогічний програмі:
*Function <ім'я> [(формальні параметри)]:<тип результату>;*
*[<розділи описів>;]
begin
<розділ операторів>
end;*
У розділі операторів повинен перебувати хоча б один оператор, що присвоює ідентифікатору функції значення. Якщо таких операторів декілька, то результатом виконання функції буде значення останнього оператора присвоювання.
Звертання до функції здійснюється за іменем з необов'язковою вказівкою списку аргументів. Кожен аргумент повинен відповідати формальним параметрам, зазначеним у заголовку, і мати той же тип.
**Формат звертання:**
*Y:=<ідентифікатор функції > [(фактичні параметри)];*
Функції можуть повертати значення цілих, дійсних, булівских, літерних і рядкових типів.
Опис **процедури** включає заголовок (ім'я) і тіло процедури. Заголовок складається з зарезервованого слова **Procedure**, ідентифікатора (імені) процедури і необов'язкового списку формальних параметрів із вказівкою їх типу, який укладається в круглі дужки .
**Формат опису:**
*Procedure <ім'я> [(формальні параметри)];*
***Приклад:***
*Procedure Korrekt;
Procedure Sort (A:byte);*
**Ім'я процедури** - ідентифікатор, унікальний у межах програми. Тіло процедури являє собою локальний блок, за структурою аналогічний програмі:
*Procedure <ім'я> [(формальні параметри)];
[<розділи описів>;]
begin
<розділи операторів>
end;*
Зверніть увагу, що як формальні параметри, так і розділ описів у процедурі може бути відсутній.
Щоб звернутися до процедури, треба використати оператор виклику процедури. Він складається з ідентифікатора (імені) процедури і списку фактичних параметрів, що відділені один від одного комами і знаходяться у круглих дужках. Якщо процедурі не передається ніяких параметрів, то фактичні параметри не вказуються.
**Формат виклику процедури:**
*<ідентифікатор> [(фактичні параметри)];*
Параметри забезпечують механізм заміни, який дозволяє виконувати процедуру з різними початковими даними. Між фактичними параметрами в операторі виклику процедури і формальними параметрами у заголовку опису процедури встановлюється взаємо-однозначна відповідність у результаті їхнього перебору зліва направо. Фактичні параметри за кількістю і типами повинні дорівнювати кількості і типам формальних параметрів.
**Параметри**, за допомогою яких здійснюється обмін значеннями змінних між підпрограмами та програмою, що їх викликає, можуть мати будь-який тип, в тому числі структурований. Існують два типи параметрів:
параметр-значення;
параметр-змінна.
Група параметрів, перед якими відсутнє зарезервоване слово*Var*, називається **параметрами-значеннями**.
Наприклад, в описі *Procedure Korrect(S,K:real) S* і *K* - параметри-значення. Формальний параметр-значення обробляється, як локальна стосовно процедури або функції, змінна. Зміни формальних параметрів-значень не впливають на відповідні значення фактичних параметрів.
Група параметрів, перед якими знаходиться ключове слово *Var*, називається **параметрами-змінними.**Наприклад, в описі *Procedure Obr(Var A,B:integer); А* та*В* - параметри-змінні. Параметр-змінна використовується в тому випадку, якщо значення повинно бути передане з процедури в блок, що її викликає. При активізації процедури або функції формальний параметр-змінна заміщується фактичною змінною, а тому будь-які зміни в значенні формального параметра-змінної відбиваються на фактичному параметрі.
І в тому, і в іншому випадку тип фактичного параметра повинен збігатися з типом формального. Якщо формальний параметр має рядковий тип, йому надається атрибут довжини, рівний 255, а тому і фактичний параметр в цьому випадку повинен також мати рядковий тип з атрибутом довжини, що дорівнює 255. У якості параметра-змінної може використовуватися будь-який тип, в тому числі файловий.
**Область дії ідентифікаторів.**
Для правильного визначення області дії ідентифікаторів при використанні в програмі процедур і функцій необхідно притримуватися наступних правил:

1. Кожний ідентифікатор повинен бути описаний перед тим, як він буде використаний.
2. Ідентифікатор діє у межах блоку, в якому він описаний.
3. Всі ідентифікатори в одному блоці повинні бути унікальними, тобто не повторюватися.
4. Однакові ідентифікатори можуть бути по-різному визначені у кожному окремому блоці, але це вважається поганим стилем програмування і тому не рекомендується в різних блоках програми використовувати змінні з однаковими іменами..
5. Якщо ідентифікатор підпрограми користувача збігається з ім'ям стандартної процедури або функції, то вони стають недоступними в межах області дії підпрограми, оголошеної користувачем, тобто стандартна функція ігнорується, а виконується програма користувача.

Наприкінці уроку рекомендуємо перевірити засвоєння матеріалу на запитаннях для самоконтролю. Наприклад:

1. Що таке підпрограма? Які типи підпрограм Ви знаєте?
2. Чим відрізняються процедури від функцій?
3. Формати опису процедур і функцій.
4. Що таке локальні та глобальні параметри? Яка між ними відмінність?
5. Які параметри називаються фактичними, а які формальними?
6. Змінні яких типів можуть використовуватись в якості параметрів процедур та функцій?
7. Чи можуть існувати підпрограми без формальних параметрів?
8. Чим відрізняються параметри-значення від параметрів-змінних?
9. Що можна ставити у відповідність формальним параметрам-значенням, а що параметрам-змінним?

Крім того, можна запропонувати учням деякі тестові завдання, наприклад, такого типу.
**Задача №415.**
***Умова***: Нехай в програмі описано такі процедури:

**Procedure** P(x,y:integer);

**Begin**

y:=x+1;

**End**;

**Procedure** Q(x:integer; **var** y:integer);

**Begin**

y:=x+1;

**End**;

**Procedure** R(**var** x,y:integer);

**Begin**

y:=x+1;

**End**;

Визначити, що буде надруковано в результаті виконання таких операторів:

|  |  |
| --- | --- |
| ***Оператори*** | ***Очікувана відповідь*** |
| c:=2; d:=1; P(c,d); writeln(d); | на екрані буде 1 |
| c:=2; d:=1; Q(c,d); writeln(d); | на екрані буде 3 |
| c:=2; d:=1; R(c,d); writeln(d); | на екрані буде 3 |

Чи припустиме таке звертання до вищезазначених процедур?

|  |  |
| --- | --- |
| ***Оператори*** | ***Очікувана відповідь*** |
| P(sqrt(c),d) | так |
| P(c,2) | так |
| Q(sqrt(c),d) | так |
| R(1,d) | ні |
| R(c,2) | ні |

Пояснення до відповідей наступні: фактичне значення параметра-значення може бути константою, змінною або виразом, а параметра-змінної - тільки іменем змінної.
**Задача №416 (1).**
***Умова:*** Знайдіть і поясніть помилки в записі функцій:

**Function** max(n:integer):real:

**Var** a,max:real;

**Begin**

**Read**(max);

**For** i:=1 **to** n-1 **do**

**Begin**

**Read**(a);

**If** a>max **then** max:=a;

**End**;

**End**;

***Очікувана відповідь учнів:***

* У функції описана внутрішня локальна змінна max, ім'я якої співпадає з іменем функції, а це неприпустимо, тому що вони обидві являються локальними для даної функції і не можуть мати однакові імена.
* Якщо все ж таки ім'я функції max, то неможливо використання його у операторах read(max) та if a>max, тому що ми отримуємо самовиклик функції, а це може призвести до помилки.
* В операторах read(max) та if a>max помилка, якщо max - ім'я функції, тому що після імені функції в момент її виклику повинні знаходитись в дужках фактичні параметри, кількість та тип яких мають співпадати з кількістю та типом фактичних параметрів даної функції (в даному випадку фактичний параметр повинен бути один).
* Якщо в програмі не існує глобальна змінна і, то вона залишається неописаною в підпрограмі.

***Домашнє завдання:***

* + прочитати сторінки 148 - 155 запропонованого підручника;
	+ задачі з запропонованого підручника №413, №414, №415(2), № 416(2).