*15. Процедури та функції. Їх описання. Глобальні та локальні імена, приклади.*

Підпрограми призначені для реалізації алгоритмів опрацювання окремих частин деякої складної задачі. Вони дають змогу реалізувати концепцію структурного програмування, суть якого полягає в розбитті складної задачі на послідовність простих підзадач і для кожної з них написання відповідних підпрограм. Підпропрограма користувача – це група команд, яку створюють в основній програмі в розділах procedure або function і до якої звертаються потрібну кількість разів з будь-якого місця програми.

Опис:

**Procedure <назва> (<список формальних параметрів>);**

**<розділ описів і оголошень процедури>**

**Begin**

**<розділ операторів процедури>**

**End;**

Функція може повертати в місце виклику лише один результат.

Опис:

**Function <назва> (<список формальних параметрів>);**

**<розділи описів і оголошень>**

**Begin**

**<Оператори>**

**Назва:=вираз**

**End;**

Змінні описані в розділі описів основної програми, називаються глобальними. Змінні, описані в розділі описів конкретної процедури називаються локальними.

Область дії ідентифікаторів – частина програми, де він може бути використаний.

**Program P;**

**Var a:integer;**

**Procedure P1;**

**Var a1:integer;**

**Procedure P2;**

**Var a2:integer;**

**begin**

**// a, a1 – global**

**// a2 – local**

**End;**

**begin**

**// a- global**

**// a1 – local**

**End;**

**Begin**

**// a – global**

**End.**

*23. Базові поняття та визначення ООП. Порядок створення об’єктів та екземплярів. Інкапсуляція, наслідування, поліморфізм. Область дії полів та методів об’єкту. Переваги ООП.*

Об’єктно-орієнтоване програмування – це методологія програмування, яка основана на представленні програми у вигляді сукупності об’єктів, кожний з яких являється реалізацією певного класу, а класи створюють ієрархію на принципах наслідування.

ООП виникло як відповідь на зростаючу складність та об’єм програм. Призначенням було дисциплінувати процес створення великих програмних комплексів і тим самим знизити їх вартість та складність.

Переваги ООП:

* Економія часу та ресурсів за рахунок використання вже написаних компонентів
* Значно менша кількість помилок в компонентах програми, що вже не раз тестувалися при використанні у інших проектах
* Можливість вести незалежну розробку програми
* Полегшення аналізу та проектування програм

Змінні об’єктних типів називаються екземплярами об’єктів, так само як записи представляють собою комбіновані структури даних.

**TPosition = Object**

**Column : integer;**

**Row : integer;**

**End;**

Обєкт характеризується як сукупність своїх властивостей так і сукупністю допустимих для даного об’єкта дій.

І структурна, і ОО-методологія мають на меті побудову ієрархічного дерева взаємозв’язків між об’єктами. Але якщо структурна ієрархія будується по принципу розділення цілого на складові частини, то при створенні ОО-ієрархії обов’язково відображається наслідування властивостей батьківських типів об’єктів дочірніми типами об’єктів.

* Наслідування – це властивість системи, що дозволяє описати новий клас на основі вже існуючих, що повністю або частково запозичує в них функціонал. Клас від якого здійснюється наслідування називається базовим, батьківським або суперкласом. Новий клас – нащадком, дочірнім або похідним класом.
* Інкапсуляція – це властивість системи, що дозволяє обєднати дані та методи, які працюють з ними в класі, та приховати деталі реалізації від користувача.
* Поліморфізм – це властивість системи використовувати об’єкти з однаковим інтерфейсом без інформації про тип та внутрішню структуру об’єкта.(Властивість різних об’єктів виконувати одну й ту саму дію по-своєму).

Область дії полів та методів:

Поля та методи поділяють одну й ту саму область дії, не зважаючи на те, що вони можуть бути описані в різних місцях тексту програми.

Якщо який-небудь екземпляр типу ObjectVar викликає метод Method, що належить йому, то виконується неявний оператор

With ObjectVar do Method

який зв’язує об’єкт і його методи в одній області дії.

**Директива private**

Директива private в рамках описання об’єкта виконує таку ж роль, як і розділ реалізації implementation в модулі. Тобто призначена для приховування деяких деталей об’єкта від користувача. Приватні поля і методи доступні тільки всередині того модуля, в якому описаний об’єкт.

**Type**

**ObjectType = object**

**Загальнодоступні поля;**

**Загальнодоступні методи;**

**private**

**Приватні поля;**

**Приватні методи;**

**End;**

**Директива public**

Директива publicробитьполя та методи загальнодоступними.

За замовчуванням усі поля та методи є загальнодоступними. Директива public дозволяє змінити порядок описання полів та методів.

**Type**

**ObjectType = object**

**private**

**Приватні поля;**

**Приватні методи;**

**public**

**Загальнодоступні поля;**

**Загальнодоступні методи;**

**End;**

**Директива protected**

Робить поля і методи доступними тільки для цього класу та нащадків цього класу, проте приховує від користувачів.

*24. Статичні та віртуальні методи. Конструктори та деструктори.*

Статичні методи – такі методи, адреса яких зв’язується з екземплярами об’єктів статично, ще під час компіляції і не змінюється до завершення роботи. Коли викликається статичний метод, для реалізації методу використовуються дані часу компіляції.

Раннє зв’язування – процес статичного зв’язування методів з реалізації об’єктів.

Пізднє зв’язування – процес динамічного зв’язування методів з реалізаціями об’єктів.

Віртуальні методи використовують пізнє зв’язування, статичні – раннє.

Синтаксично віртуальні методи відрізняються від динамічних наявністю директиви virtual. Віртуальні методи оптимізовані по швидкодії, а динамічні – по об’єму коду.

Якщо оголошується віртуальний метод в батьківському типі, то це накладає обмеження на його дочірні типи:

* Усі однойменні методи дочірніх типів також повинні бути віртуальними
* Заголовки одного методу повинні бути ідентичними, включно з числом параметрів і їх типів
* Кожен об’єкт з віртуальними методами повинен мати конструктор

Конструктор – це спеціальний тип процедури, яка виконує початкові установки для роботи механізму спеціальних методів.

Кожен окремий екземпляр об’єкту має бути ініційований окремим викликом конструктора.

Деструктор – спеціальний тип процедури, «збірник сміття». Призначений для видалення динамічно розміщенних об’єктів та звільнення памяті, що вони займають.

Деструктори можуть бути як статичними, так і віртуальними. Їх можна наслідувати.

*25. Динамічні об’єкти. Помилки, що виникають при роботі з динамічними об’єктами. Директиви public та private.*

Динамічними називають такі програмні об’єкти, які виникають в процесі виконання програми.

Аналогічно динамічним змінним будь-яких інших типів пам'ять під динамічні об’єкти виділяється процедурами New і Dispose:

**Var**

**PWKg, PBKg : ^TKing;**

При використанні динамічних об’єктів зручно працювати з розширеним синтаксисом процедур New і Dispose.

Перше розширення дозволяє викликати New як функцію:

**Type**

**PTKing = ^TKing;**

**Var**

**PWKg, PBKg : PRKing;**

**Begin**

**PWKg := New(PTKing);**

**PBKg := New(PTKing);**

**…**

**End.**

Друге розширення дозволяє використовувати другий параметр, який вказує конструктору імя конструктора з фактичними параметрами, що ініціює динамічний об’єкт відразу ж після виділення для нього область памяті.

**New (PWKg, Init (‘e’, 1, WhiteColor ));**

**PWKg^.Display;**

Розширення Dispose аналогічно другому розширенню процедури New. Відмінність полягає в тому, що другий параметр є імя не конструктора, а деструктора.

**Dispose(PWKg, Done);**

Помилки під час роботи з динамічними об’єктами

1. Під час виклику процедури New с двома параметрами імя конструктора не може бути складеним, адже під звернення динамічний об’єкт ще не створено.

Наприклад оператор

**New(Pline, PLine^.Init);** видасть повідомлення про помилку

2. Якщо не вистачає памяті для розміщення екземпляра динамічного об’єкта, то виклик конструктора генерує фатальну помилку з кодом 203.

HeapFunc – функція, що дозволяє обробляти помилки динамічно розподіленої області памяті.

**function HeapFunc(Size: word): integer; far;**

Ця фунція реалізується за допомогою присвоєння її адреси змінній HeapError:

**HeapError := @HeapFunc;**

Якщо параметр **Size** містить розмір блоку, для якого не знайшлося області памяті відповідного розміру, то функція обробки памяті спробує звільнити блок, розмір якого не менше.

**Директива private**

Директива private в рамках описання об’єкта виконує таку ж роль, як і розділ реалізації implementation в модулі. Тобто призначена для приховування деяких деталей об’єкта від користувача. Приватні поля і методи доступні тільки всередині того модуля, в якому описаний об’єкт.

**Type**

**ObjectType = object**

**Загальнодоступні поля;**

**Загальнодоступні методи;**

**private**

**Приватні поля;**

**Приватні методи;**

**End;**

**Директива public**

Директива publicробитьполя та методи загальнодоступними.

За замовчуванням усі поля та методи є загальнодоступними. Директива public дозволяє змінити порядок описання полів та методів.

**Type**

**ObjectType = object**

**private**

**Приватні поля;**

**Приватні методи;**

**public**

**Загальнодоступні поля;**

**Загальнодоступні методи;**

**End;**

**Директива protected**

Робить поля і методи доступними тільки для цього класу та нащадків цього класу, проте приховує від користувачів.

*26. Концепція RAD. Система розробки програм Delphi та її IDE.*

RAD (*rapid application development*) – це концепція створення засобів розробки програмних продуктів, що виділяє особливу увагу на швидкість і зручність програмування, створенню технологічного процесу, що дозволяє програмісту максимально щвидко створювати програми.

Ця концепція виникла в кінці 20 століття, як відповідь на незграбні методи програмування 70-80 років, передбачали настільки повільний процес створення програми, що навіть вимоги до програми встигали змінюватися ще до завершення її розробки. RAD передбачає, що розробка ПЗ здійснюється невеликою командою протягом 3-4 місяців з використання візуального програмування. Однією з особливостей є залучення замовника вже на ранніх стадіях розробки.

Основні принципи:



* Створення *прототипу* для уточнення вимог замовника
* Циклічність розробки: кожна нова версія продукту базується на оцінці результата роботи попередньої версії програми замовником
* Мінімізація часу розробки версії за рахунок переносу готових модулів і додавання функціоналу в нову версію
* Тісна співпраця команди розробників
* Інструментарій повинен бути націлений на мінімізацію часу розробки

Delphi – це інтегрована середа швидкої розробки програмного забезпечення для Microsoft Windows, Mac OS, iOS и Android на мові Delphi(Object Pascal). Створена фірмою Borland, зараз належить Embarcadero Technologies.

Завдяки унікальному поєднанню простоти мови та генерації машинного коду дозволяє безпосередньо та на досить низькому рівні взаємодіяти з операційною системою та бібліотеками.

Середовище Delphi

* Головне меню
* Панель інструментів
* Палітра компонентів
* Інспектор об’єктів
* Вікно форми
* Редактор коду