Лукаш І.М.
ЧДПУ імені Т.Г.Шевченка

Контейнерні типи даних

Провідною лінією сучасного шкільного курсу інформатики визнається “інформаційне моделювання”. До визначення поняття “інформаційна модель” існує різні підходи. Один з них визначає інформаційні моделі, як “клас знакових моделей, що описують інформаційні процеси (виникнення, передачу, перетворювання та використання інформації) в системах різної природи”[1]. Тобто, з точки зору програмування під побудовою інформаційної моделі можна розуміти створення в програмному коді абстрактних типів даних, зокрема структурних (масивів, записів, об’єктів тощо) при описанні реальної системи, яка задається умовою задачі. Наприклад, інформацію про людину можна описати за допомогою об’єктного типу даних, який як атрибути буде містити дані про її прізвище, ім’я, стать, дату народження, посаду і т.д., а як методи – алгоритми визначення віку людини, кількості діб до дня народження, розміру заробітної плати тощо. Цей тип даних дозволяє мати кілька окремих екземплярів, конкретних людей (учнів класу, співробітників компанії, читачів в бібліотеці), які згідно з умовою деякої задачі треба розглядати як одне ціле. Звісно, що можна скористатися таким типом даних як масив, але при розв’язуванні певних задач виникатимуть додаткові труднощі з реалізацією алгоритмів вставки елемента в середину масиву (запис читача), вилучення елемента з масиву (звільнення робітника) тощо. Запобігти цих труднощів дозволяють контейнерні типи даних: списки, стеки, черги, колекції. Якщо вивченню масивів, записів, об’єктів в шкільному курсі інформатики приділяється певна увага, то контейнерні типи даних залишаються поза увагою учнів, хоч вони важливі для побудови інформаційних моделей реальних систем тому, що дозволяють описувати відношення “N-арної асоціації” між об’єктами. Наприклад, між об’єктами абонент і читач - абонемент бібліотеки містить інформацію про читачів, кількість яких постійно змінюється.

Такі абстрактні типи даних, як стеки і черги походять від відповідних механізмів для тимчасового збереження даних, реалізованих в мові асемблера. Стек працює згідно з принципом LIFO (“Last In, Fist Out” – “останнім прийшов, першим вийшов”), черга – згідно з принципом FIFO (“Fist In, Fist Out” – “перший прийшов, перший вийшов”). Список як список динамічних структурних даних можна реалізувати вже в мові Turbo Pascal. Пізніше удосконалений механізм списку був реалізований у вигляді колекцій в Turbo Vision для мови Pascal. В системі візуального програмування Delphi зазначеними механізмами збереження та обробки даних можна користуватися як стандартними об’єктними типами даних, що містяться в модулі Contnrs: TClassList - список класів, TComponentList – список компонент, TObjectList – список об’єктів, TObjectQueue – черга об’єктів, TObjectStack – стек об’єктів, TOderedList – упорядкований список (запис, зчитування елементів можна

здійснювати тільки або з початку, або з кінця), TQueue – черга покажчиків, TStack – стек покажчиків, TList – список покажчиків, TCollection – "візуальний" список об'єктів (використовуються при створенні компонент).

Розглянемо еволюцію такого абстрактного типу даних, як клас TList. Ознайомлення учнів зі списками, зокрема лінійними, є доцільним. Зокрема, вони використовуються при розв'язуванні окремих олімпіадних задач, наприклад, для реалізації розв'язків на основі графової моделі (знаходження шляхів Гамільтона та Ейлера), які розглядаються на факультативних заняттях з інформатики. Так, знаходження гамільтонова шляху (обхід всіх вершин графа по одному разу) передбачає на основі матриці досяжності застосування алгоритму вставки вершин. Згідно з цим алгоритмом нова вершина, що добавляється до шляху, вставляється між двома вершинами попередньо сформованого списку вершин шляху, з якими вона має зв'язок, або ставиться на початок чи кінець цього списку. Якщо на якомусь етапі вставка чергової вершини неможлива, то гамільтонів шлях не існує. Переважна частина учнів при розв'язуванні олімпіадних задач віддають перевагу мові Turbo Pascal. Зрозуміло, що розв'язування зазначеної задачі із застосуванням списку значно спрощує роботу. Окрім цього, розуміння реалізації механізмів роботи з елементами списку в Turbo Pascal, робить зрозумілішим подальше сприйняття методів класу TList в Delphi. Тому доцільно спочатку розглянути основні операції з даними списку в Turbo Pascal.

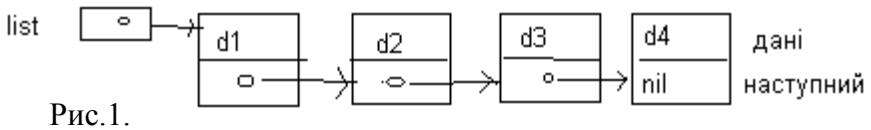

Рис.1.

Кожний елемент списку є записом, що складається з двох частин: даних і покажчика на наступний елемент списку. Кінець списку помічається покажчиком nil. Початок списку формує покажчик list, який вказує на перший елемент списку (рис.1). До операцій роботи зі списком належать такі: додавання елемента до списку, вилучення елемента зі списку, пошук елемента в списку, виведення даних про всі або окремі елементи списку.

Оголошення списку як абстрактного типу даних:

```
Type
     datetype=integer;
      Z=^elementtype;
      Elementtype= record
                       Dd: datetype;
                       Next: z end;
Var list:z; d:datetype;
```

Елемент списку задається за допомогою запису Elementtype, а сам тип списку (Z) – як покажчик на цей елемент списку.

Формування списку, додавання чергового елемента до списку, елемент ставиться в кінець списку:

```
Procedure InList(var l:z; d:datetype);
{В кінець списку, що починається з l, добавляється елемент, який містить в якості частини даних d. Покажчик p переглядає список зверху донизу, поки не знайде nil. }
var p,q:z;
begin
new(q); q^.next:=nil; q^.Dd:=d;
{Формування елемента, що буде добавлятися в кінець списку. Тепер будемо відшукувати кінець списку}
if l=nil then i:=q {Список пустий}
         else  begin
          p:=l;
       while P^.next<>nil do
     p:=p^.next;{переміщаємо покажчик}
      p^.next:=q;{добавляємо елемент}
end;
```

Додавання елемента до списку, новий елемент вставляється між двома існуючими елементами списку:

```
procedure InList1(var l:z; x,d:datetype);
{В список, що починається з l , після елемента, що містить дане x, добавляється елемент, що містить d}
var q:z;
begin
  if l=nil then writeln('Елемент не знайдений')
           else
if l^.Dd<>x then InList(l^.next,x,d)
         else begin new(q);
              q^.Dd:=d;
              q^.next:=l^.next;
             l^.next:=q;
          end;
end;
```

Виведення даних про всі елементи списку:

```
procedure PrintList(l:z);
{рекурсивна процедура роздрукування списку}
begin
if l=nil then writeln('Кінець списку')
        else  begin writeln(l^.Dd);
           PrintList(l^.next);
           End;
End;
```

Пошук елемента із заданою властивістю в списку, його вилучення:

```
Procedure OutList(l:z, d:datetype);
{Вилучити елемент з даними d зі списку }
var q:z;{Покажчик, що вказується на елемент, який вилучається}
begin
 if l=nil then writeln('Елемент не знайдений')
           else
```

```
if l^.data<> d then OutList(l^.next,d)
      else begin q=1; {перевести покажчик з шуканого елемента на сторонній}
            l:=l^.next;{покажчик попереднього елемента перевести на наступний}
           dispose(q);{Звільнити ячейку пам'яті}
        end;
end;
```

Як бачимо, в мові програмування Turbo Pascal для роботи зі списком, зокрема лінійними, структурними елементами якого є записи, доводиться писати спеціальні підпрограми. Порівняно з Turbo Pascal в Delphi надається можливість формувати списки не тільки із записів, а й списки, елементи яких є класи, об'єкти (причому *в одному списку можуть об'єднуватися об'єкти різних класів*). Виконання різних операцій зі списком і його елементами здійснюється за допомогою спеціальних механізмів без написання відповідного програмного коду, як це робилося в Turbo Pascal. Абстрактні типи TComponentList (список компонент), TClassList (список класів) та TObjectList (список об'єктів) є нащадками (прямими чи непрямими) від класу TList, тобто успадковують всі його методи роботи зі списком. Тому достатньо розглянути їх на прикладі одного з наведених класів, наприклад, TObjectList, з яким важливо ознайомитись учням для подальшого його застосування при реалізації відношення "N-арної асоціації" між об'єктами.

Елементи класу позначаються TObjectList як Items[index:integer]:TObject та починають нумеруватися з 0. Як вже зазначалось, оголошення цього класу містить в модулі Contnrs, тобто його необхідно підключати до програми, в якій передбачається робота зі списками об'єктів (Uses Contnrs). Клас TObjectList є нащадком від класу TList (для створення списку із записів), який в свою чергу є нащадком від класу TObject. У зв'язку з цим для роботи зі списком та його елементами клас TObjectList має як свої власні методи, так і наслідувані від батьківських класів. Ті, що найчастіше використовуються наведені в Таблиці 1, з іншими можна ознайомитись, скориставшись Help додатком Delphi.

Таблиця 1.

Методи класу TObjectList.

Назва методу	Формат методу	Призначення методів
Create	Constructor Create	Створення в пам'яті нового списку
Add	Function Add(A:TObject):integer	Добавлення елемента до списку. Перший індекс елемента в списку має індекс 0. Як результат виконання метод надає індекс добавленого до списку елемента.
Count	Function Count:integer;	Кількість елементів у списку
Clear	procedure Clear;	Очищення списку від елементів. Елементи звільняються з пам'яті.
Destroy	destructor Destroy;	Ліквідація списку з пам'яті (руйнація списку)

Delete	procedure Delete(Index: Integer);	Вилучення зі списку елемента, що стоїть на позиції Index. Вилучений елемент звільняється з пам'яті. Всі наступні елементи зсуваються на один індекс вперед.
Extract	function Extract(Item: Pointer): Pointer;	Вилучення елемента із списку. Елемент не звільняється із пам'яті.
Remove	function Remove (AObject: TObject): Integer;	Вилучення останнього елемента заданого класу без вказування його індексу. Вилучений об'єкт руйнується, а наступні елементи зсуваються на один індекс вверх.
IndexOf	function IndexOf (AObject: TObject): Integer;	Знаходження індексу останнього елемента заданого класу в списку.
Insert	procedure Insert(Index: Integer; AObject: TObject);	Вставити елемент в список перед вказаною позицією.
First	function First: Pointer;	Повертає Items[0] – перший елемент списку.
Last	function Last: Pointer;	Повертає Items[Count-1] – останній елемент списку.
Pack	procedure Pack;	Вилучення пустих (nil) елементів зі списку
Exchange	Procedure Exchange (Index1, Index2: Integer);	Поміняти елементи місцями

Принципи роботи з лінійним списком об'єктів розглянемо на прикладі списку, що складається з об'єктів Книжок. Окремий об'єкт Книга характеризується прізвищем автора, назвою книги, її ціною. В конструкторі об'єкта Книга передбачимо введення значень атрибутів з клавіатури.

Оголошення об'єкта:

```
Uses Contnrs; {Підключаємо модуль для роботи з класом TObjectList}
Type
TBook=class {Оголошення класу Книга}
FAuthor,FTitle:string; {Атрибути: прізвище автора та назва книги}
FPrice:real; {Атрибут ціна книги}
Constructor Create; {Перевизначення конструктора}
Property Author:string read FAuthor write FAuthor;
{Властивість, що здійснює доступ до атрибута прізвище автора}
Property Title:string read FTitle write FTitle;
{Властивість, що здійснює доступ до атрибута назва книги}
Property Price:real read FPrice write FPrice;
{Властивість, що здійснює доступ до атрибута ціна книги}
End;
Var A:TBook; L:TObjectList;i:integer; {Описання екземплярів об'єктів та змінних}
Constructor TBook.Create; {Реалізація конструктора об'єкта Книга}
Begin
Inherited Create;
Writeln('Введіть інформацію про книгу');
Write('Автор');readln(FAuthor);
Write('Назва книги');readln(FTitle);
```

```
Write('Ціна');readln(FPrice);
End;
```

Далі, за допомогою наступного фрагмента програми розглянемо реалізацію операцій роботи з лінійним списком (L) об'єктів класу TBook.

Сформуємо список, наприклад, із трьох книжок:

```
begin
L:=TObjectList.Create; {Створення в пам'яті об'єкта список}
For i:=1 to 3 do
Begin
A:=TBook.Create; {Створення чергового елемента книга}
L.Add(A); {Добавити створений об'єкт до списку}
End;
writeln(L.Count); {Роздрукування кількості елементів у списку}
For i:=0 to L.Count-1 do {Роздрукування назв книжок, що занесені до списку}
Begin
Writeln(TBook(L.Items[i]).Title); {Звернення до окремого елемента списку}
End;
L.Clear; {Вилучення об'єктів зі списку та із пам'яті}
L.Destroy; {Руйнування списку}
end.
```

Вхідним параметром багатьох методів є змінна типу TObject. У зв'язку з тим, що всі об'єкти Delphi походять від стандартного класу TObject, то згідно з принципом сумісності класів (кожному екземпляру батьківського класу можна присвоїти екземпляр класу-нащадка, але не навпаки) на місце формального параметру типу TObject в методах класу можна підставити екземпляр будь-якого класу, створеного програмістом. У зворотному випадку, коли із списку нам необхідно прочитати його елемент, змінній типу класу-нащадка не можна присвоїти елемент типу батьківського класу, тому наступне присвоювання призводить до помилки несумісності типів: A:=L.Items[i]. Доступ до окремого елемента списку, до його методів здійснюється через вказання типу класу: TBook(L.Items[i]).Title . Якщо в списку передбачається наявність об'єктів різних класів, то перед доступом до окремого об'єкта необхідно робити перевірку, до якого класу він належить за допомогою оператора **is** (є елементом класу):

```
if TObject(L.Items[i]) is TBook then   Writeln(TBook(L.Items[i]).Price);
```

Інші операції роботи зі списком та його елементами виконуються аналогічно, наприклад:

вилучення першого елемента зі списку: L.Delete(1);

вставка нового екземпляра книжки на другу позицію у списку:

```
A:=TBook.Create; L.Insert(2,A);
```

вилучення елемента без його руйнування в пам'яті:

```
A:=L.Extract(L.Items[1]); writeln(A.Price);
```

отримання індексу останнього елемента заданого класу: i:=L.Indexof(A);

вилучення зі списку останнього елемента заданого класу: L.Remove(A);

доступ до першого елемента списку: A:=L.First; writeln(A.Price);.

Наведені приклади свідчать про доступнішу для учнів, простішу роботу зі списками в Delphi порівняно з Turbo Pascal. В Delphi такі складні операції, як вставка елемента, його вилучення і т.д. зводиться до виклику методу звичайного класу.

Інші контейнерні типи мають власні методи опрацювання своїх елементів. Так класи TStack, TQueue та їх похідні мають методи:
Peek – в стеку переводить покажчик на вершину стеку, в черзі встановлює покажчик на межу черги, де знаходиться перший (раніший) елемент;
Pop – вилучення верхнього елемента в стеку або першого елемента в черзі;
Push – додає елемент до вершини стеку або додає елемент в кінець черги.

Колекція – це об'єкт, що містить як властивість список інших об'єктів, і застосовується при створенні компонент. Наприклад, стандартна компонента Delphi для малювання простих графічних зображень Shape має властивість Shape, яка дозволяє задавати вид графічного зображення (прямокутник, еліпс тощо). Тобто ця властивість містить список об'єктів – окремих геометричних фігур. Нехай необхідно створити новий компонент типу Shape з більшою кількістю геометричних зображень. Для цього треба буде спочатку створити окремі елементи – об'єкти зображень. Потім із сукупності цих об'єктів сформувати колекцію, яка як властивість включається у створювану компоненту. Сама колекція є нащадком класу TCollection. Кожний елемент, який вона містить, - нащадок від класу TCollectionItem. Зазначені класи є базовими для створення колекції, мають певну функціональність, необхідну для неї. Так клас TCollectionItem має, наприклад, такі властивості: Collection – вказує на колекцію, якій належить елемент; DisplayName – ім'я класу елемента; ID – унікальний целочисельний ідентифікатор елемента в середині колекції; Index – порядковий номер елемента в колекції. Клас TCollection має як сходні з класом TList особливості, так і відмінні: Count – кількість елементів в колекції; ItemClass – дає фактичний клас елементів колекції (всі елементи колекції мають бути одного класу, який задається при створенні колекції, і надалі не може бути зміненим); Add, Clear, Insert – аналоги відповідних методів класу TList; Assign – метод для копіювання елементів із однієї колекції до іншої; FindItemID – дає елемент колекції із заданим ID.

Як показують розглянуті в статті приклади та описання контейнерних типів даних, їх застосування дозволяє значно розширити можливості розробників інформаційних моделей.

Література

1. Могилев А.В., Хеннер Е.К. О понятиии «Информационное моделирование»// Информатика и образование - №8, 1997. – С.3-7.