МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

Юго-Западный государственный университет

(ЮЗГУ)

Кафедра вычислительной техники

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«__» _____ 2022 г.

ПРОЕКТИРОВАНИЕ КЛАССА-ОБЕРТКИ ДЛЯ ОДНОМЕРНОГО МАССИВА

Методические указания по выполнению лабораторной работы
для студентов направления подготовки 09.03.01

Курск 2022

УДК 621.3

Составитель: Э.И. Ватутин

Рецензент
Кандидат технических наук, доцент *Т.Н. Конаныхина*

Проектирование класса-обертки для одномерного массива: методические указания по выполнению лабораторной работы по дисциплинам «Программирование», «Объектно-ориентированное программирование» / Юго-Зап. гос. ун-т; сост.: Э.И. Ватутин; Курск, 2022. 9 с.: ил. 1.

Методические рекомендации содержат сведения по разработке программ с использованием объектно-ориентированной технологии на современных языках программирования высокого уровня.

Предназначены для студентов направления подготовки 09.03.01 «Информатика и вычислительная техника».

Текст печатается в авторской редакции

Подписано в печать _____. Формат 60x84 1/16.
Усл. печ. л. Уч. – изд.л. Тираж 30 экз. Заказ 315. Бесплатно.
Юго-Западный государственный университет
305040, Курск, ул. 50 лет Октября, 94.

Содержание

Проектирование абстрактного базового класса

Целью работы является проектирование класса-обертки над одномерным массивом, инкапсулирующего в своем составе базовые действия, выполняемые над массивами:

- добавление элементов;
- удаление элементов;
- поиск заданного элемента;
- обращение к элементам по индексу.

Каждое действие снабжается рядом проверок, контролирующих его корректность (например, отсутствие ошибок, связанных с выходом за пределы массива). Соответствующий абстрактный класс представлен ниже.

```
TOneDymArr = class
private
  function GetItemsCnt(): Integer; virtual; abstract;

  function GetItem(Index: Integer): Integer; virtual; abstract;
  procedure SetItem(Index: Integer; Value: Integer); virtual; abstract;
public
  procedure AddItem(Value: Integer); virtual; abstract;
  procedure DeleteItem(Value: Integer); virtual; abstract;

  function Find(Value: Integer): Integer; virtual; abstract;

  function ConvertToStr(): String; virtual; abstract;

  property ItemsCnt: Integer read GetItemsCnt;
  property Items[Index: Integer]: Integer read GetItem write SetItem;
end;
```

Он включает в своем составе методы `AddItem` и `DeleteItem` для добавления/удаления элементов, метод `Find` для поиска элемента по значению, а также свойства `ItemsCnt` и `Items`, позволяющие определить текущее число элементов в массиве и безопасно обращаться к ним по индексу.

Проектирование класса-наследника для хранения данных на базе одномерного статического массива

Для организации хранения данных на базе одномерного статического массива создадим класс наследник, в котором необходимо переопределить виртуальные абстрактные методы класса-предка и создать поля для хранения данных (одномерный статический массив) и текущего числа хранимых элементов.

```
const
  MAX_LENGTH = 1000;

type
  TOneDymStaticArr = class(TOneDymArr)
  private
    fArr: array [1..MAX_LENGTH] of Integer;      { Одномерный статический
                                                   массив }
    fCurrCnt: Integer;                           { Текущее число хранимых
                                                   элементов }

    function GetItemsCnt(): Integer; override;

    function GetItem(Index: Integer): Integer; override;
    procedure SetItem(Index: Integer; Value: Integer); override;
  public
    constructor Create();

    procedure AddItem(Value: Integer); override;
    procedure DeleteItem(Value: Integer); override;

    function Find(Value: Integer): Integer; override;

    function ConvertToString(): String; override;
  end;

function TOneDymStaticArr.GetItem(Index: Integer): Integer;
begin
  ASSERT((Low(fArr) <= Index) and (Index <= High(fArr)), 'Выход за пределы
      индексов массива');

  Result := fArr[Index];
end;

procedure TOneDymStaticArr.SetItem(Index, Value: Integer);
begin
  ASSERT((Low(fArr) <= Index) and (Index <= High(fArr)), 'Выход за пределы
      индексов массива');

  fArr[Index] := Value;
end;

constructor TOneDymStaticArr.Create();
begin
  fCurrCnt := 0;
end;
```

```
procedure TOneDymStaticArr.AddItem(Value: Integer);
begin
  Inc(fCurrCnt);

  ASSERT(fCurrCnt <= High(fArr), 'Массив переполнен');

  fArr[fCurrCnt] := Value;
end;

function TOneDymStaticArr.ConvertToString(): String;
var
  I: Integer;
begin
  Result := '[';

  for I := 1 to fCurrCnt do
    Result := Result + IntToStr(fArr[I]) + ', ';

  if Length(Result) > 2 then
    SetLength(Result, Length(Result)-2);

  Result := Result + ']';
end;

function TOneDymStaticArr.Find(Value: Integer): Integer;
var
  I: Integer;
begin
  for I := 1 to fCurrCnt do
    if fArr[I] = Value then begin
      Result := I;
      exit;
    end;

  Result := -1;
  end;

procedure TOneDymStaticArr.DeleteItem(Value: Integer);
var
  Index: Integer;
begin
  Index := Find(Value);

  if Index < 0 then
    exit;

  fArr[Index] := fArr[fCurrCnt];
  Dec(fCurrCnt);
end;

function TOneDymStaticArr.GetItemsCnt(): Integer;
begin
  Result := fCurrCnt;
end;

var
  Arr: TOneDymArr;


begin
  { Пример использования статического массива }
  Arr := TOneDymStaticArr.Create();

  { Добавление элементов }
```

```
  Arr.AddItem(1);
  Arr.AddItem(2);
  Arr.AddItem(4);
  Arr.AddItem(9);
  Writeln(Arr.ConvertToString());

  { Удаление несуществующего элемента }
  Arr.DeleteItem(3);
  Writeln(Arr.ConvertToString());

  { Удаление существующего элемента }
  Arr.DeleteItem(2);
  Writeln(Arr.ConvertToString());

  Readln;
end.
```

Результат работы программы приведен на рис. 1.

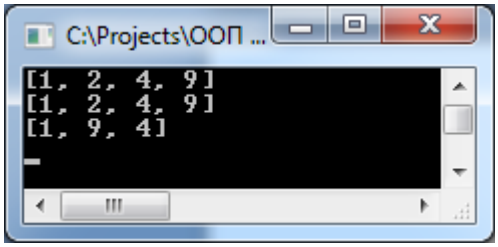

Рис. 1. Результат работы с созданным классом на базе одномерного статического массива

Аналогичным образом, переопределяя методы классов `TOneDymArr` и `TOneDymStaticArr` с использованием виртуального полиморфизма, можно добиться различного поведения класса, сохраняя идентичным набор методов для работы с ним. Подобным образом можно добиться контроля дублирования значений (фактически реализовать функционал множества или мультимножества), упорядоченного хранения данных с целью реализации операции быстрого бинарного поиска, использования различных структур данных (двоичные деревья, кучи, динамические списки), позволяющих более быструю практическую реализацию тех или иных действий, при этом детали реализации работы с данными будут скрыты от программиста, использующего функционал, предоставляемый разработанными классами.

Индивидуальные задания

В соответствии с индивидуальным вариантом реализовать требуемый функционал путем разработки соответствующего класса-наследника от рассмотренных выше классов. Разработать программу, подтверждающую корректность работы методов разработанного класса.

1. Хранение не упорядоченных данных без дублирования на базе статического массива.
2. Хранение упорядоченных по возрастанию данных на базе статического массива.
3. Хранение не упорядоченных данных без дублирования на базе динамического массива.
4. Хранение упорядоченных по убыванию данных без дублирования на базе динамического массива.
5. Хранение не упорядоченных данных в виде односвязного динамического списка.
6. Хранение упорядоченных по возрастанию данных в виде односвязного динамического списка.
7. Хранение не упорядоченных данных в виде двухсвязного динамического списка.
8. Хранение упорядоченных по убыванию данных в виде двухсвязного динамического списка.
9. Хранение не упорядоченных данных в виде односвязного динамического кольца.
10. Хранение упорядоченных по возрастанию данных в виде односвязного динамического кольца.
11. Хранение не упорядоченных данных в виде двухсвязного динамического кольца.
12. Хранение упорядоченных по убыванию данных в виде двухсвязного динамического кольца.
13. Хранение данных в виде бинарного дерева.

Контрольные вопросы

1. Какие 3 основные парадигмы объектно-ориентированного программирования существуют?
2. Для чего применяется наследование?
3. В чем отличие классов от объектов?
4. Какие модификаторы доступа к компонентам классов существуют?
5. В чем отличие статического и виртуального полиморфизма?
6. Что такое свойства и чем они отличаются от полей?
7. В чем отличие метаклассов от классов?
8. Что такое конструктор и деструктор?

Библиографический список

1. Емельянов С.Г., Ватутин Э.И., Панищев В.С., Титов В.С. Процедурно-модульное программирование на Delphi: учебное пособие. М.: Аргамак-Медиа, 2014. 352 с.
2. Зотов И.В., Ватутин Э.И., Борзов Д.Б. Процедурно-ориентированное программирование на C++: учебное пособие. Курск: КурскГТУ, 2008. 211 с.