```
 1: program PascalOOP;
 2:
 3: (* Программа: Pascal "Объекты"
 4:  * Выполнил: _______
 5:  *
 6:  * При разработке программы применена технология
 7:  * объектно-ориентированного программирования (ООП).
 8:  * Построены два класса: TPostalAddress, позволяющий
 9:  * работать с информацией о почтовом адресе организации
10:  * и TMainMenu, реализующий меню для выполнения всех возможных
11:  * действий с объектом первого класса. В объектах
12:  * используются динамические поля.
13:  *)
14:
15:
16: uses
17:   Crt; { стандартный модуль Turbo Pascal для использования функций:
18:          * ReadKey - подождать нажатие клавиши и вернуть её символ
19:          * ClrScr - очистить экран }
20:
21:
22: type
23:   (* TPostalAddress - класс, содержащий информацию
24:    * о почтовом адресе организации
25:    *)
26:   TPostalAddress = object
27:     { Поля класса: }
28:     Country: ^string; { Страна }
29:     Index: ^integer;  { Индекс }
30:     City: ^string;    { Город }
31:     Street: ^string;  { Улица }
32:     HouseNumber: ^string; { Номер дома, string для записей вида '23A' }
33:
34:     constructor Create; { конструктор для создания объекта класса }
35:     destructor Destroy; { деструктор для уничтожения объекта класса }
36:
37:     procedure Print; { печатает текущие значения полей объекта }
38:
39:     (* методы для чтения/установки соответствующих полей объекта: *)
40:     function GetCountry: string;
41:     procedure SetCountry(Value: string);
42:     function GetIndex: integer;
43:     procedure SetIndex(Value: integer);
44:     function GetCity: string;
45:     procedure SetCity(Value: string);
46:     function GetStreet: string;
47:     procedure SetStreet(Value: string);
48:     function GetHouseNumber: string;
49:     procedure SetHouseNumber(Value: string);
50:   end;
51:
52:   (* TMainMenu - класс главного меню программы
53:    *)
54:   TMainMenu = object
55:     { используемый для работы объект "почтовый адрес" }
56:     PostalAddress: ^TPostalAddress;
57:
58:     constructor Create; { конструктор для создания объекта класса }
59:     destructor Destroy; { деструктор для уничтожения объекта класса }
60:
61:     procedure StartDialog; { запустить меню в режиме диалога }
62:   end;
63:
64:
65: { TPostalAddress }
66:
67: constructor TPostalAddress.Create;
68: begin
69:   (* обнуляем поля объекта: *)
70:   Country:= nil;
71:   Index:= nil;
```

```
 72:   City:= nil;
 73:   Street:= nil;
 74:   HouseNumber:= nil;
 75:
 76:   WriteLn('Объект класса TPostalAddress создан!');
 77: end;
 78:
 79: destructor TPostalAddress.Destroy;
 80: begin
 81:   (* Очищаем память, занимаемую динамическими полями объекта: *)
 82:   if Country <> nil then Dispose(Country);
 83:   if Index <> nil then Dispose(Index);
 84:   if City <> nil then Dispose(City);
 85:   if Street <> nil then Dispose(Street);
 86:   if HouseNumber <> nil then Dispose(HouseNumber);
 87:   (* присваиваем nil для поддержания хорошего стиля программирования *)
 88:   Country:= nil;
 89:   Index:= nil;
 90:   City:= nil;
 91:   Street:= nil;
 92:   HouseNumber:= nil;
 93:
 94:   WriteLn('Объект класса TPostalAddress уничтожен!');
 95: end;
 96:
 97: function TPostalAddress.GetCity: string;
 98: begin
 99:   GetCity:= City^;
100: end;
101:
102: function TPostalAddress.GetCountry: string;
103: begin
104:   GetCountry:= Country^;
105: end;
106:
107: function TPostalAddress.GetHouseNumber: string;
108: begin
109:   GetHouseNumber:= HouseNumber^;
110: end;
111:
112: function TPostalAddress.GetStreet: string;
113: begin
114:   GetStreet:= Street^;
115: end;
116:
117: function TPostalAddress.GetIndex: integer;
118: begin
119:   GetIndex:= Index^;
120: end;
121:
122: procedure TPostalAddress.SetCity(Value: string);
123: begin
124:   New(City);       { отводим память для динамического поля }
125:   City^:= Value;   { присваиваем значение }
126: end;
127:
128: procedure TPostalAddress.SetCountry(Value: string);
129: begin
130:   New(Country);    { отводим память для динамического поля }
131:   Country^:= Value; { присваиваем значение }
132: end;
133:
134: procedure TPostalAddress.SetHouseNumber(Value: string);
135: begin
136:   New(HouseNumber); { отводим память для динамического поля }
137:   HouseNumber^:= Value; { присваиваем значение }
138: end;
139:
140: procedure TPostalAddress.SetStreet(Value: string);
141: begin
142:   New(Street);    { отводим память для динамического поля }
```

```
143:   Street^:= Value; { присваиваем значение }
144: end;
145:
146: procedure TPostalAddress.SetIndex(Value: integer);
147: begin
148:   New(Index);      { отводим память для динамического поля }
149:   Index^:= Value; { присваиваем значение }
150: end;
151:
152: (* Функция для перевода integer в string *)
153: function IntToStr(Value: integer): string;
154: var
155:   s: string;
156: begin
157:   str(Value, s);
158:   IntToStr:= s;
159: end;
160:
161: procedure TPostalAddress.Print;
162: begin
163:   WriteLn('* Содержимое полей PostalAddress: *');
164:
165:   if Country <> nil then WriteLn('Country = ', Country^)
166:   else WriteLn('Country = nil (динамическое поле не задано)');
167:
168:   if Index <> nil then WriteLn( 'Index = ', IntToStr(Index^) )
169:   else WriteLn('Index = nil (динамическое поле не задано)');
170:
171:   if City <> nil then WriteLn('City = ', City^)
172:   else WriteLn('City = nil (динамическое поле не задано)');
173:
174:   if Street <> nil then WriteLn('Street = ', Street^)
175:   else WriteLn('Street = nil (динамическое поле не задано)');
176:
177:   if HouseNumber <> nil then WriteLn('HouseNumber = ', HouseNumber^)
178:   else WriteLn('HouseNumber = nil (динамическое поле не задано)');
179:
180:   WriteLn;
181: end;
182:
183: { TMainMenu }
184:
185: constructor TMainMenu.Create;
186: begin
187:   PostalAddress:= nil;
188:
189:   WriteLn('Объект класса TMainMenu создан!');
190: end;
191:
192: destructor TMainMenu.Destroy;
193: begin
194:   if PostalAddress <> nil then begin
195:     Dispose(PostalAddress, Destroy);
196:     PostalAddress:= nil;
197:   end;
198:
199:   WriteLn('Объект класса TMainMenu уничтожен!');
200: end;
201:
202: procedure TMainMenu.StartDialog;
203: var
204:   Done: boolean; { пользователь хочет выйти? }
205:   Key: char;     { нажатая клавиша}
206:   s: string;     { для ввода адреса - элемент строка }
207:   i: longint;    { для ввода адреса - элемент число }
208: begin
209:   Done:= false;
210:   repeat
211:     WriteLn;
212:     WriteLn('1: Создать объект PostalAddress (с пустыми полями)');
213:     WriteLn('2: Создать и заполнить объект PostalAddress');
```

```
214:     WriteLn('3: Уничтожить объект');
215:     WriteLn('4: ! Напечатать содержимое полей объекта !');
216:     WriteLn('5: Напечатать страну');
217:     WriteLn('6: Изменить страну');
218:     WriteLn('7: Напечатать индекс');
219:     WriteLn('8: Изменить индекс');
220:     WriteLn('9: Напечатать город');
221:     WriteLn('0: Изменить город');
222:     WriteLn('D: Напечатать улицу');
223:     WriteLn('F: Изменить улицу');
224:     WriteLn('H: Напечатать номер дома');
225:     WriteLn('J: Изменить номер дома');
226:     WriteLn('Q: ! Выход из программы !');
227:     WriteLn;
228:     WriteLn;
229:
230:     Key:= UpCase(ReadKey);
231:     case Key of
232:       (* 1: Создать объект (с пустыми полями) *)
233:       '1': begin
234:         if PostalAddress <> nil
235:         then WriteLn('Ошибка: Объект уже создан! Вначале уничтожте существующий!')
236:         else New(PostalAddress, Create);
237:       end;
238:       (* 2: Создать и заполнить объект *)
239:       '2': begin
240:         if PostalAddress <> nil
241:         then WriteLn('Ошибка: Объект уже создан! Вначале уничтожте существующий!')
242:         else begin
243:           New(PostalAddress, Create);
244:           (* Ввод данных *)
245:           WriteLn('Введите страну: ');
246:           ReadLn(s);
247:           PostalAddress^.SetCountry(s);
248:
249:           WriteLn('Введите индекс: ');
250:           ReadLn(i);
251:           PostalAddress^.SetIndex(i);
252:
253:           WriteLn('Введите город: ');
254:           ReadLn(s);
255:           PostalAddress^.SetCity(s);
256:
257:           WriteLn('Введите улицу: ');
258:           ReadLn(s);
259:           PostalAddress^.SetStreet(s);
260:
261:           WriteLn('Введите номер дома: ');
262:           ReadLn(s);
263:           PostalAddress^.SetHouseNumber(s);
264:
265:           WriteLn('Завершено!');
266:         end;
267:       end;
268:       (* 3: Уничтожить объект *)
269:       '3': begin
270:         if PostalAddress = nil
271:         then WriteLn('Ошибка: Объект не создан! Вначале создайте объект!')
272:         else begin
273:           Dispose(PostalAddress, Destroy);
274:           PostalAddress:= nil;
275:         end;
276:       end;
277:       (* 4: ! Напечатать содержимое полей объекта ! *)
278:       '4': begin
279:         if PostalAddress = nil then WriteLn('Ошибка: Объект не создан!')
280:         else PostalAddress^.Print;
281:       end;
282:
283:       (* 5: Напечатать страну *)
284:       '5': begin
```

```
285:         if PostalAddress = nil then begin
286:           WriteLn('Ошибка: Объект не создан! Вначале создайте объект!');
287:           break;
288:         end;
289:         if PostalAddress^.Country <> nil
290:         then WriteLn('PostalAddress^.Country = ', PostalAddress^.Country^)
291:         else WriteLn('PostalAddress^.Country = nil (динамическое поле не задано)');
292:       end;
293:       (* 7: Напечатать индекс *)
294:       '7': begin
295:         if PostalAddress = nil then begin
296:           WriteLn('Ошибка: Объект не создан! Вначале создайте объект!');
297:           break;
298:         end;
299:   if PostalAddress^.Index <> nil
300:   then WriteLn( 'PostalAddress^.Index = ', IntToStr(PostalAddress^.Index^) )
301:   else WriteLn('PostalAddress^.Index = nil (динамическое поле не задано)');
302:       end;
303:       (* 9: Напечатать город *)
304:       '9': begin
305:         if PostalAddress = nil then begin
306:           WriteLn('Ошибка: Объект не создан! Вначале создайте объект!');
307:           break;
308:         end;
309:   if PostalAddress^.City <> nil
310:   then WriteLn('PostalAddress^.City = ', PostalAddress^.City^)
311:   else WriteLn('PostalAddress^.City = nil (динамическое поле не задано)');
312:       end;
313:       (* D: Напечатать улицу *)
314:       'D': begin
315:         if PostalAddress = nil then begin
316:           WriteLn('Ошибка: Объект не создан! Вначале создайте объект!');
317:           break;
318:         end;
319:   if PostalAddress^.Street <> nil
320:   then WriteLn('PostalAddress^.Street = ', PostalAddress^.Street^)
321:   else WriteLn('PostalAddress^.Street = nil (динамическое поле не задано)');
322:       end;
323:       (* H: Напечатать номер дома *)
324:       'H': begin
325:         if PostalAddress = nil then begin
326:           WriteLn('Ошибка: Объект не создан! Вначале создайте объект!');
327:           break;
328:         end;
329:   if PostalAddress^.HouseNumber <> nil
330:   then WriteLn('PostalAddress^.HouseNumber = ', PostalAddress^.HouseNumber^)
331:   else WriteLn('PostalAddress^.HouseNumber = nil (динамическое поле не задано)');
332:       end;
333:
334:       (* 6: Изменить страну *)
335:       '6': begin
336:         if PostalAddress = nil then begin
337:           WriteLn('Ошибка: Объект не создан! Вначале создайте объект!');
338:           break;
339:         end;
340:
341:         WriteLn('Введите страну: ');
342:         ReadLn(s);
343:         PostalAddress^.SetCountry(s);
344:
345:         WriteLn('Поле обновлено!');
346:       end;
347:       (* 8: Изменить индекс *)
348:       '8': begin
349:         if PostalAddress = nil then begin
350:           WriteLn('Ошибка: Объект не создан! Вначале создайте объект!');
351:           break;
352:         end;
353:
354:         WriteLn('Введите индекс: ');
355:         ReadLn(i);
```

```
356:         PostalAddress^.SetIndex(i);
357:
358:         WriteLn('Поле обновлено!');
359:       end;
360:       (* 0: Изменить город *)
361:       '0': begin
362:         if PostalAddress = nil then begin
363:           WriteLn('Ошибка: Объект не создан! Вначале создайте объект!');
364:           break;
365:         end;
366:
367:         WriteLn('Введите город: ');
368:         ReadLn(s);
369:         PostalAddress^.SetCity(s);
370:
371:         WriteLn('Поле обновлено!');
372:       end;
373:       (* F: Изменить улицу *)
374:       'F': begin
375:         if PostalAddress = nil then begin
376:           WriteLn('Ошибка: Объект не создан! Вначале создайте объект!');
377:           break;
378:         end;
379:
380:         WriteLn('Введите улицу: ');
381:         ReadLn(s);
382:         PostalAddress^.SetStreet(s);
383:
384:         WriteLn('Поле обновлено!');
385:       end;
386:       (* J: Изменить номер дома *)
387:       'J': begin
388:         if PostalAddress = nil then begin
389:           WriteLn('Ошибка: Объект не создан! Вначале создайте объект!');
390:           break;
391:         end;
392:
393:         WriteLn('Введите номер дома: ');
394:         ReadLn(s);
395:         PostalAddress^.SetHouseNumber(s);
396:
397:         WriteLn('Поле обновлено!');
398:       end;
399:
400:       (* Q: Выход из программы *)
401:       'Q': begin
402:         if PostalAddress <> nil then begin
403:           Dispose(PostalAddress, Destroy);
404:           PostalAddress:= nil;
405:         end;
406:         Done:= true;
407:       end;
408:     end;
409:
410:     if not Done then begin
411:       WriteLn('нажмите любую клавишу...');
412:       ReadKey;
413:       ClrScr;
414:     end;
415:   until Done;
416: end;
417:
418:
419: (******************* MAIN ****************************************)
420:
421: var
422:   MainMenu: ^TMainMenu; { объект "главное меню" }
423:
424: begin
425:   ClrScr; { очистить экран }
426:   WriteLn('Программа запущена!');
```

```
427:   New(MainMenu, Create); { создаём объект MainMenu }
428:
429:   WriteLn('Запуск меню...');
430:   MainMenu^.StartDialog; { Вывод меню и начало работы }
431:
432:   WriteLn('Завершается работа программы...');
433:   Dispose(MainMenu, Destroy); { уничтожаем объект MainMenu }
434:
435:   WriteLn('Нажмите любую клавишу...');
436:   ReadKey;
437: end.
```