|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

|  |  |
| --- | --- |
| **ФАКУЛЬТЕТ** | **ИУК «Информатика и управление»** |
| **КАФЕДРА** | **ИУК2 «Информационные системы и сети»** |
|  | |

**Практическая работа 3.1**

**«****Примеры стандартных решений в С#»**

**ДИСЦИПЛИНА: «Объектно-ориентированное программирование»**



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Выполнил: студент гр. ИУК4-21Б | |  |  | ( | Суриков Н.С | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |
| Проверил: | |  |  | ( | Дерюгина Е. О. | ) |
|  |  |  | (подпись) |  | (Ф.И.О.) |  |

|  |  |
| --- | --- |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | |
|  | - Балльная оценка:  - Оценка: |

**Цель:** сформировать навыки решения задач с применением ООП.

**Листинг:**

1 public class Program

2 {

3 *// 1. Переписать числа в обратном порядке*

4 static void ReverseNumbersInFile(string inputFileName, string

5 outputFileName)

6 {

7 Stack<string> numbersStack = new Stack<string>();

8 using (StreamReader sr = new StreamReader(inputFileName))

9 {

10 while (!sr.EndOfStream)

11 {

12 string line = sr.ReadLine();

13 numbersStack.Push(line);

14 }

15 }

16 using (StreamWriter sw = new StreamWriter(outputFileName))

17 {

18 while (numbersStack.Count > 0)

19 {

20 sw.WriteLine(numbersStack.Pop());

21 }

22 }

23 }

24 static void PrintReversedVowels(string fileName)

25 {

26 Stack<char> vowelsStack = new Stack<char>();

27 using (StreamReader sr = new StreamReader(fileName))

28 {

29 while (!sr.EndOfStream)

30 {

31 char c = (char)sr.Read();

32 if (IsVowel(c))

33 {

34 vowelsStack.Push(c);

35 }

36 }

37 }

38 while (vowelsStack.Count > 0)

39 {

40 Console.Write(vowelsStack.Pop());

41 }

42 Console.WriteLine();

43 }

44 *// 3. Напечатать литеры каждой строки в обратном порядке*

45 static void PrintCharactersInReverseOrder(string fileName)

46 {

47 using (StreamReader sr = new StreamReader(fileName))

48 {

49 while (!sr.EndOfStream)

50 {

51 string line = sr.ReadLine();

52 char[] chars = line.ToCharArray();

53 Array.Reverse(chars);

54 Console.WriteLine(chars);

55 }

56 }

57 }

58 *// Вспомогательная функция для определения гласных букв*

59 static bool IsVowel(char c)

60 {

61 return "aeiouAEIOU".IndexOf(c) != -1;

62 }

63 *// 4. Проверка, является ли строка s2 обратной s1*

64 static bool IsReverseString(string s1, string s2)

65 {

66 if (s1.Length != s2.Length)

67 return false;

68 Stack<char> stack = new Stack<char>();

69 foreach (char c in s1)

70 {

71 stack.Push(c);

72 }

73 foreach (char c in s2)

74 {

75 if (stack.Count == 0 || stack.Pop() != c)

76 return false;

77 }

78 return true;

79 }

80 *// 5. Подсчет выражения в префиксной форме*

81 static int EvaluatePrefixExpression(string expression)

82 {

83 Stack<int> stack = new Stack<int>();

84 for (int i = expression.Length - 1; i >= 0; i--)

85 {

86 char c = expression[i];

87 if (char.IsDigit(c))

88 {

89 stack.Push(c - '0');

90 }

91 else if (c == '+' || c == '-' || c == '\*' || c == '/')

92 {

93 int operand1 = stack.Pop();

94 int operand2 = stack.Pop();

95 switch (c)

96 {

97 case '+':

98 stack.Push(operand1 + operand2);

99 break;

100 case '-':

101 stack.Push(operand1 - operand2);

102 break;

103 case '\*':

104 stack.Push(operand1 \* operand2);

105 break;

106 case '/':

107 stack.Push(operand1 / operand2);

108 break;

109 }

110 }

111 }

112 return stack.Pop();

113 }

114 *// 6. Преобразование выражения из инфиксной формы в префиксную*

115 static string ConvertInfixToPrefix(string infixExpression)

116 {

117 Stack<string> operatorStack = new Stack<string>();

118 Stack<string> operandStack = new Stack<string>();

119 string[] tokens = infixExpression.Split(' ');

120 for (int i = tokens.Length - 1; i >= 0; i--)

121 {

122 string token = tokens[i];

123 if (IsOperator(token))

124 {

125 while (operatorStack.Count > 0 &&

126 Precedence(operatorStack.Peek()) >= Precedence(token))

127 {

128 string operand1 = operandStack.Pop();

129 string operand2 = operandStack.Pop();

130 operandStack.Push(token + " " + operand1 + " " +

131 operand2);

132 operatorStack.Pop();

133 }

134 operatorStack.Push(token);

135 }

136 else

137 {

138 operandStack.Push(token);

139 }

140 }

141 while (operatorStack.Count > 0)

142 {

143 string operand1 = operandStack.Pop();

144 string operand2 = operandStack.Pop();

145 operandStack.Push(operatorStack.Pop() + " " + operand1 + " " +

146 operand2);

147 }

148 return operandStack.Pop();

149 }

150 *// Проверка является ли символ оператором*

151 static bool IsOperator(string token)

152 {

153 return token == "+" || token == "-" || token == "\*" || token == "/";

154 }

155 *// Определение приоритета оператора*

156 static int Precedence(string op)

157 {

158 switch (op)

159 {

160 case "+":

161 case "-":

162 return 1;

163 case "\*:":

164 case "/":

165 return 2;

166 default:

167 return 0;

168 }

169 }

170 *// 7. Преобразование выражения из постфиксной формы в инфиксную*

171 static string ConvertPostfixToInfix(string postfixExpression)

172 {

173 Stack<string> stack = new Stack<string>();

174 string[] tokens = postfixExpression.Split(' ');

175 foreach (string token in tokens)

176 {

177 if (IsOperator(token))

178 {

179 string operand2 = stack.Pop();

180 string operand1 = stack.Pop();

181 stack.Push("(" + operand1 + " " + token + " " + operand2 +

182 ")");

183 }

184 else

185 {

186 stack.Push(token);

187 }

188 }

189 return stack.Pop();

190 }

191 *// 8. Вычисление значения формулы из файла*

192 static int EvaluateFormulaFromFile(string fileName)

193 {

194 Stack<char> operators = new Stack<char>();

195 Stack<int> values = new Stack<int>();

196 using (StreamReader sr = new StreamReader(fileName))

197 {

198 string formula = sr.ReadLine();

199 foreach (char c in formula)

200 {

201 if (c == '(')

202 {

203 continue;

204 }

205 else if (char.IsDigit(c))

206 {

207 values.Push(int.Parse(c.ToString()));

208 }

209 else if (c == 'm' || c == 'M')

210 {

211 operators.Push(c);

212 }

213 else if (c == ')' && operators.Count > 0)

214 {

215 int val1 = values.Pop();

216 int val2 = values.Pop();

217 int result;

218 if (operators.Pop() == 'M')

219 {

220 result = Math.Max(val1, val2);

221 }

222 else

223 {

224 result = Math.Min(val1, val2);

225 }

226 values.Push(result);

227 }

228 }

229 }

230 return values.Pop();

231 }

232 *// 9. Преобразование формулы из файла по заданному правилу*

233 static int EvaluateFormulaWithRules(string fileName)

234 {

235 Stack<char> operators = new Stack<char>();

236 Stack<int> values = new Stack<int>();

237 using (StreamReader sr = new StreamReader(fileName))

238 {

239 string formula = sr.ReadLine();

240 foreach (char c in formula)

241 {

242 if (c == '(')

243 {

244 continue;

245 }

246 else if (char.IsDigit(c))

247 {

248 values.Push(int.Parse(c.ToString()));

249 }

250 else if (c == 'm' || c == 'p')

251 {

252 operators.Push(c);

253 }

254 else if (c == ')' && operators.Count > 0)

255 {

256 int val1 = values.Pop();

257 int val2 = values.Pop();

258 int result;

259 if (operators.Pop() == 'm')

260 {

261 result = (val2 - val1) % 10;

262 }

263 else

264 {

265 result = (val2 + val1) % 10;

266 }

267 values.Push(result);

268 }

269 }

270 }

271 return values.Pop();

272 }

273 *// 10. Преобразование текста с учетом символа "#"*

274 static string ProcessTextWithBackspace(string text)

275 {

276 Stack<char> stack = new Stack<char>();

277 foreach (char c in text)

278 {

279 if (c != '#')

280 {

281 stack.Push(c);

282 }

283 else if (stack.Count > 0)

284 {

285 stack.Pop();

286 }

287 }

288 char[] resultChars = stack.ToArray();

289 Array.Reverse(resultChars);

290 return new string(resultChars);

291 }

292 *// 11. Печать элементов файла: сначала все символы, отличные от цифр, затем все цифры*

293 static void PrintFileElements(string fileName)

294 {

295 Queue<char> nonDigitChars = new Queue<char>();

296 Queue<char> digitChars = new Queue<char>();

297 using (StreamReader sr = new StreamReader(fileName))

298 {

299 while (!sr.EndOfStream)

300 {

301 char c = (char)sr.Read();

302 if (char.IsDigit(c))

303 {

304 digitChars.Enqueue(c);

305 }

306 else

307 {

308 nonDigitChars.Enqueue(c);

309 }

310 }

311 }

312 while (nonDigitChars.Count > 0)

313 {

314 Console.Write(nonDigitChars.Dequeue());

315 }

316 while (digitChars.Count > 0)

317 {

318 Console.Write(digitChars.Dequeue());

319 }

320 Console.WriteLine();

321 }

322 *// 12. Печать элементов файла: сначала числа из интервала [a,b], затем числа меньше a, затем числа больше b*

323 static void PrintFileElementsInRange(string fileName, int a, int b)

324 {

325 Queue<int> numbersInRange = new Queue<int>();

326 Queue<int> numbersLessThanA = new Queue<int>();

327 Queue<int> numbersGreaterThanB = new Queue<int>();

328 using (StreamReader sr = new StreamReader(fileName))

329 {

330 while (!sr.EndOfStream)

331 {

332 int number = int.Parse(sr.ReadLine());

333 if (number >= a && number <= b)

334 {

335 numbersInRange.Enqueue(number);

336 }

337 else if (number < a)

338 {

339 numbersLessThanA.Enqueue(number);

340 }

341 else

342 {

343 numbersGreaterThanB.Enqueue(number);

344 }

345 }

346 }

347 while (numbersInRange.Count > 0)

348 {

349 Console.WriteLine(numbersInRange.Dequeue());

350 }

351 while (numbersLessThanA.Count > 0)

352 {

353 Console.WriteLine(numbersLessThanA.Dequeue());

354 }

355 while (numbersGreaterThanB.Count > 0)

356 {

357 Console.WriteLine(numbersGreaterThanB.Dequeue());

358 }

359 }

360 *// 13. Печать слов из файла: сначала слова, начинающиеся на гласную, затем на согласную*

361 static void PrintWordsByFirstLetter(string fileName)

362 {

363 Queue<string> vowelsWords = new Queue<string>();

364 Queue<string> consonantsWords = new Queue<string>();

365 using (StreamReader sr = new StreamReader(fileName))

366 {

367 while (!sr.EndOfStream)

368 {

369 string word = sr.ReadLine();

370 if (!string.IsNullOrEmpty(word))

371 {

372 char firstChar = word[0];

373 if (IsVowel(firstChar))

374 {

375 vowelsWords.Enqueue(word);

376 }

377 else

378 {

379 consonantsWords.Enqueue(word);

380 }

381 }

382 }

383 }

384 while (vowelsWords.Count > 0)

385 {

386 Console.WriteLine(vowelsWords.Dequeue());

387 }

388 while (consonantsWords.Count > 0)

389 {

390 Console.WriteLine(consonantsWords.Dequeue());

391 }

392 }

393 *// 14. Печать чисел из файла: сначала положительные числа, затем отрицательные числа*

394 static void PrintNumbersBySign(string fileName)

395 {

396 Queue<int> positiveNumbers = new Queue<int>();

397 Queue<int> negativeNumbers = new Queue<int>();

398 using (StreamReader sr = new StreamReader(fileName))

399 {

400 while (!sr.EndOfStream)

401 {

402 int number = int.Parse(sr.ReadLine());

403 if (number >= 0)

404 {

405 positiveNumbers.Enqueue(number);

406 }

407 else

408 {

409 negativeNumbers.Enqueue(number);

410 }

411 }

412 }

413 while (positiveNumbers.Count > 0)

414 {

415 Console.WriteLine(positiveNumbers.Dequeue());

416 }

417 while (negativeNumbers.Count > 0)

418 {

419 Console.WriteLine(negativeNumbers.Dequeue());

420 }

421 }

422 *// 15. Печать слов из файла: сначала слова с прописной буквы, затем со строчной*

423 static void PrintWordsByCase(string fileName)

424 {

425 Queue<string> capitalizedWords = new Queue<string>();

426 Queue<string> lowercasedWords = new Queue<string>();

427 using (StreamReader sr = new StreamReader(fileName))

428 {

429 while (!sr.EndOfStream)

430 {

431 string word = sr.ReadLine();

432 if (!string.IsNullOrEmpty(word))

433 {

434 if (char.IsUpper(word[0]))

435 {

436 capitalizedWords.Enqueue(word);

437 }

438 else

439 {

440 lowercasedWords.Enqueue(word);

441 }

442 }

443 }

444 }

445 while (capitalizedWords.Count > 0)

446 {

447 Console.WriteLine(capitalizedWords.Dequeue());

448 }

449 while (lowercasedWords.Count > 0)

450 {

451 Console.WriteLine(lowercasedWords.Dequeue());

452 }

453 }

454 *// 16. Печать данных о сотрудниках: сначала данные о мужчинах, затем о женщинах*

455 static void PrintEmployeesByGender(string fileName)

456 {

457 Queue<string> maleEmployees = new Queue<string>();

458 Queue<string> femaleEmployees = new Queue<string>(); using (StreamReader sr = new StreamReader(fileName))

459 {

460 while (!sr.EndOfStream)

461 {

462 string employeeData = sr.ReadLine();

463 if (!string.IsNullOrEmpty(employeeData))

464 {

465 string[] data = employeeData.Split(',');

466 string gender = data[3].Trim();

467 if (gender.ToLower() == "male")

468 {

469 maleEmployees.Enqueue(employeeData);

470 }

471 else if (gender.ToLower() == "female")

472 {

473 femaleEmployees.Enqueue(employeeData);

474 }

475 }

476 }

477 }

478 while (maleEmployees.Count > 0)

479 {

480 Console.WriteLine(maleEmployees.Dequeue());

481 }

482 while (femaleEmployees.Count > 0)

483 {

484 Console.WriteLine(femaleEmployees.Dequeue());

485 }

486 }

487 *// 17. Печать данных о сотрудниках: сначала данные о сотрудниках с зарплатой меньше 10000, затем остальные*

488 static void PrintEmployeesBySalary(string fileName)

489 {

490 Queue<string> lowSalaryEmployees = new Queue<string>();

491 Queue<string> otherEmployees = new Queue<string>();

492 using (StreamReader sr = new StreamReader(fileName))

493 {

494 while (!sr.EndOfStream)

495 {

496 string employeeData = sr.ReadLine();

497 if (!string.IsNullOrEmpty(employeeData))

498 {

499 string[] data = employeeData.Split(',');

500 int salary = int.Parse(data[5].Trim());

501 if (salary < 10000)

502 {

503 lowSalaryEmployees.Enqueue(employeeData);

504 }

505 else

506 {

507 otherEmployees.Enqueue(employeeData);

508 }

509 }

510 }

511 }

512 while (lowSalaryEmployees.Count > 0)

513 {

514 Console.WriteLine(lowSalaryEmployees.Dequeue());

515 }

516 while (otherEmployees.Count > 0)

517 {

518 Console.WriteLine(otherEmployees.Dequeue());

519 }

520 }

521 *// 18. Печать данных о сотрудниках: сначала данные о сотрудниках младше 30 лет, затем остальные*

522 static void PrintEmployeesByAge(string fileName)

523 {

524 Queue<string> youngEmployees = new Queue<string>();

525 Queue<string> otherEmployees = new Queue<string>();

526 using (StreamReader sr = new StreamReader(fileName))

527 {

528 while (!sr.EndOfStream)

529 {

530 string employeeData = sr.ReadLine();

531 if (!string.IsNullOrEmpty(employeeData))

532 {

533 string[] data = employeeData.Split(',');

534 int age = int.Parse(data[4].Trim());

535 if (age < 30)

536 {

537 youngEmployees.Enqueue(employeeData);

538 }

539 else

540 {

541 otherEmployees.Enqueue(employeeData);

542 }

543 }

544 }

545 }

546 while (youngEmployees.Count > 0)

547 {

548 Console.WriteLine(youngEmployees.Dequeue());

549 }

550 while (otherEmployees.Count > 0)

551 {

552 Console.WriteLine(otherEmployees.Dequeue());

553 }

554 }

555 *// 19. Печать данных о студентах: сначала данные о студентах, успешно сдавших сессию, затем остальные*

556 static void PrintStudentsBySessionResult(string fileName)

557 {

558 Queue<string> passedStudents = new Queue<string>();

559 Queue<string> otherStudents = new Queue<string>();

560 using (StreamReader sr = new StreamReader(fileName))

561 {

562 while (!sr.EndOfStream)

563 {

564 string studentData = sr.ReadLine();

565 if (!string.IsNullOrEmpty(studentData))

566 {

567 string[] data = studentData.Split(',');

568 int mark1 = int.Parse(data[4].Trim());

569 int mark2 = int.Parse(data[5].Trim());

570 int mark3 = int.Parse(data[6].Trim());

571 if (mark1 >= 4 && mark2 >= 4 && mark3 >= 4)

572 {

573 passedStudents.Enqueue(studentData);

574 }

575 else

576 {

577 otherStudents.Enqueue(studentData);

578 }

579 }

580 }

581 }

582 while (passedStudents.Count > 0)

583 {

584 Console.WriteLine(passedStudents.Dequeue());

585 }

586 while (otherStudents.Count > 0)

587 {

588 Console.WriteLine(otherStudents.Dequeue());

589 }

590 }

591 *// 20. Печать данных о студентах: сначала данные о студентах, успешно обучающихся на оценки 4 и 5, затем остальные*

592 static void PrintStudentsByGoodMarks(string fileName)

593 {

594 Queue<string> goodStudents = new Queue<string>();

595 Queue<string> otherStudents = new Queue<string>();

596 using (StreamReader sr = new StreamReader(fileName))

597 {

598 while (!sr.EndOfStream)

599 {

600 string studentData = sr.ReadLine();

601 if (!string.IsNullOrEmpty(studentData))

602 {

603 string[] data = studentData.Split(',');

604 int mark1 = int.Parse(data[4].Trim());

605 int mark2 = int.Parse(data[5].Trim());

606 int mark3 = int.Parse(data[6].Trim());

607 if (mark1 == 4 || mark1 == 5 && mark2 == 4 || mark2 == 5

608 && mark3 == 4 || mark3 == 5)

609 {

610 goodStudents.Enqueue(studentData);

611 }

612 else

613 {

614 otherStudents.Enqueue(studentData);

615 }

616 }

617 }

618 } while (goodStudents.Count > 0)

619 {

620 Console.WriteLine(goodStudents.Dequeue());

621 }

622 while (otherStudents.Count > 0)

623 {

624 Console.WriteLine(otherStudents.Dequeue());

625 }

626 }

627 static void Main(string[] args)

628 {

629 *// Вызов функций для решения задач с использованием класса Stack*

630 Console.WriteLine("Задачи с использованием класса Stack:");

631 Console.WriteLine();

632 *// 1. Переписать в другой файл все числа в обратном порядке*

633 ReverseNumbersInFile("input.txt", "output.txt");

634 *// 2. Распечатать гласные буквы текстового файла в обратном порядке*

635 PrintReversedVowels("text.txt");

636 *// 3. Напечатать содержимое текстового файла, выписывая литеры каждой строки в обратном порядке*

637 PrintCharactersInReverseOrder("text.txt");

638 *// 4. Проверка строки s2 на обратность по отношению к s1*

639 string s1 = "hello";

640 string s2 = "olleh";

641 bool isReverse = IsReverseString(s1, s2);

642 Console.WriteLine($"Строка s2 обратна строке s1: {isReverse}");

643 *// 5. Вычисление выражения в префиксной форме*

644 string prefixExpression = "+ 3 \* 4 5";

645 int prefixResult = EvaluatePrefixExpression(prefixExpression);

646 Console.WriteLine($"Результат выражения в префиксной форме:{prefixResult}");

647 *// 6. Преобразование выражения из инфиксной формы в префиксную*

648 string infixExpression = "3 + 4 \* 5";

649 string prefixExpressionConverted = ConvertInfixToPrefix(infixExpression);

650 Console.WriteLine($"Префиксная форма выражения:{prefixExpressionConverted}");

651 *// 7. Преобразование выражения из постфиксной формы в инфиксную*

652 string postfixExpression = "3 4 5 \* +";

653 string infixExpressionConverted =

654 ConvertPostfixToInfix(postfixExpression);

655 Console.WriteLine($"Инфиксная форма выражения:{infixExpressionConverted}");

656 *// 8. Вычисление значения формулы из текстового файла*

657 int formulaValue = EvaluateFormulaFromFile("formula.txt");

658 Console.WriteLine($"Значение формулы: {formulaValue}");

659 *// 9. Вычисление значения формулы с операциями m и p из текстового файла*

660 int formulaValue2 = EvaluateFormulaWithRules("formula2.txt");

661 Console.WriteLine($"Значение формулы с операциями m и p:{formulaValue2}");

662 *// 10. Преобразование текста с учетом символа "#"*

663 string processedText = ProcessTextWithBackspace("abc#d##c");

664 Console.WriteLine($"Обработанный текст: {processedText}");

665 Console.WriteLine();

666 Console.WriteLine("Задачи с использованием класса Queue:");

667 Console.WriteLine();

668 *// Вызов функций для решения задач с использованием класса Queue*

669 *// 11. Печать элементов файла: сначала все символы, отличные от цифр, затем все цифры*

670 PrintFileElements("text.txt");

671 *// 12. Печать элементов файла: сначала числа из интервала [a,b], затем числа меньше a, затем числа больше b*

672 PrintFileElementsInRange("numbers.txt", 3, 7);

673 *// 13. Печать слов из файла: сначала слова, начинающиеся на гласную, затем на согласную*

674 PrintWordsByFirstLetter("words.txt");

675 *// 14. Печать чисел из файла: сначала положительные числа, затем отрицательные числа*

676 PrintNumbersBySign("numbers.txt");

677 *// 15. Печать слов из файла: сначала слова с прописной буквы, затем со строчной*

678 PrintWordsByCase("words.txt");

679 *// 16. Печать данных о сотрудниках: сначала данные о мужчинах, затем о женщинах*

680 PrintEmployeesByGender("employees.txt");

681 *// 17. Печать данных о сотрудниках: сначала данные о сотрудниках с зарплатой меньше 10000, затем остальные*

682 PrintEmployeesBySalary("employees.txt");

683 *// 18. Печать данных о сотрудниках: сначала данные о сотрудниках младше 30 лет, затем остальные*

684 PrintEmployeesByAge("employees.txt");

685 *// 19. Печать данных о студентах: сначала данные о студентах, успешно сдавших сессию, затем остальные*

686 PrintStudentsBySessionResult("students.txt");

687 *// 20. Печать данных о студентах: сначала данные о студентах, успешно обучающихся на оценки 4 и 5, затем остальные*

688 PrintStudentsByGoodMarks("students.txt");

689 Console.ReadLine();

690 }

691 }

**Вывод:** в результате работы мы получили навыки использования решения задач с использованием стандартных коллекций на языке C#.

**Основная литература**

1. Зыков, С. В. Введение в теорию программирования. Объектно-ориентированный подход : учебное пособие / С. В. Зыков. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 187 c. — ISBN 978-5-4497-0926-4. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/102007.html.

2. Павловская, Т. А. Программирование на языке высокого уровня C# : учебное пособие / Т. А. Павловская. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 245 c. — Текст : электронный — URL: http://www.iprbookshop.ru/102051.html.

3. Биллиг, В. А. Основы объектного программирования на C# (С# 3.0, Visual Studio 2008) : учебник / В. А. Биллиг. — 3-е изд. — Москва : Интернет-Университет Информационных Технологий (ИНТУИТ), Ай Пи Ар Медиа, 2021. — 409 c. — Текст : электронный — URL: http://www.iprbookshop.ru/102029.html.

4. Горелов, С. В. Современные технологии программирования: разработка Windows-приложений на языке С#. В 2 томах. Т.I : учебник / С. В. Горелов ; под редакцией П. Б. Лукьянова. — Москва : Прометей, 2019. — 362 c. — Текст : электронный — URL: http://www.iprbookshop.ru/94532.html.

5. Горелов, С. В. Современные технологии программирования: разработка Windows-приложений на языке С#. В 2 томах. Т.II : учебник / С. В. Горелов ; под редакцией П. Б. Лукьянова. — Москва : Прометей, 2019. — 378 c. — Текст : электронный — URL: http://www.iprbookshop.ru/94533.html.