|  |  |
| --- | --- |
| http://www.pl130.ru/doc/index/%D0%9B%D0%BE%D0%B3%D0%BE%D1%82%D0%B8%D0%BF22.jpg | Санкт-Петербургское государственное бюджетное  профессиональное образовательное учреждение  "Колледж электроники и приборостроения" |

отчет

**по практической работе № 15**

**по дисциплине «Основы алгоритмизации и программирования»**

Тема: Программирование модуля

|  |  |  |
| --- | --- | --- |
| Студент гр. 13 ИТ |  | Липинский К.С. |
| Преподаватель |  | Счастливцев А.К. |

Санкт-Петербург

2024

**Цель работы.**

Обучение разделению программы на модули для облегчения понимания кода.

**Выполнение работы.**

Program.cs:

namespace Prac;  
public partial class Program  
{  
 private static void Main()  
 {  
 Console.Write("Choose task: ");  
 char choice = Convert.ToChar(Console.ReadLine()!);  
   
 Chooser(choice);  
 }  
}

Chooser.cs:

namespace Prac;  
public partial class Program  
{  
 private static void Chooser(char choice)  
 {  
 switch (choice)  
 {  
 case 'C':  
 C();  
 break;  
 case 'D':  
 D();  
 break;  
 case 'E':  
 E();  
 break;  
 case 'F':  
 F();  
 break;  
 case 'G':  
 G();  
 break;  
 default:  
 Console.WriteLine("An error occured!");  
 break;  
 }  
 }  
}

Папка Tasks:

C.cs:

namespace Prac;  
using static Math;  
  
public partial class Program  
{  
 private static void C()  
 {  
 int x = Convert.ToInt32(Console.ReadLine());  
 double a = Sin(x);  
 double z = Sqrt(Abs(Sin(x) / (Pow(x, 2) + 4)) + Sin(Pow(a, 2)));  
  
 Console.WriteLine(z);  
 }  
}

D.cs:

namespace Prac;  
using static Math;  
  
public partial class Program  
{  
 private static void D()  
 {   
 int x = Convert.ToInt32(Console.ReadLine());  
 double a = 0.5;  
 double y = Log(Sqrt(Sin(x) / 5 \* x + 4)) - Pow(Double.**E**, a / x) + Sqrt(Sin(x / Pow(x, 2) + 1));  
   
 Console.WriteLine(y);  
 }  
}

E.cs:

namespace Prac;  
using static Math;  
  
public partial class Program  
{  
 private static void E()  
 {  
 int x = Convert.ToInt32(Console.ReadLine());  
 int n = Convert.ToInt32(Console.ReadLine());  
 int counter = n;  
 double s = 0;  
 while (counter > 0)  
 {  
 counter--;  
 s += 1 / Pow(x, counter);  
 }  
 Console.WriteLine(s);  
 }  
}

F.cs:

namespace Prac;  
  
public partial class Program  
{  
 private static void F()  
 {  
 int[] x = new int [20];  
 for (int i = 0; i < x.Length; i++)  
 {  
 Random rand = new();  
 x[i] = rand.Next(0, 50);  
 }  
 int s = 0;  
 int m = 0;  
 for (int i = 0; i < x.Length; i++)  
 {  
 if (i / 2 == 0)  
 {  
 s += x[i];  
 m++;  
 }  
 }  
 Console.WriteLine(s / m);  
 }  
}

G.cs:

namespace Prac;  
  
public partial class Program  
{  
 private static void G()  
 {  
 int y = Convert.ToInt32(Console.ReadLine());  
 int[,] x = new int [10, 10];  
 for (int i = 0; i < x.GetLength(0); i++)  
 {  
 for (int j = 0; j < x.GetLength(1); j++)  
 {  
 Random rand = new();  
 x[i, j] = rand.Next(1, 30);  
 }  
 }  
 for (int i = 0; i < x.GetLength(0); i++)  
 {  
 for (int j = 0; j < x.GetLength(1); j++)  
 {  
 if (i + j < y && (i + j) % 2 == 0)  
 {  
 x[i, j] = 0;  
 }  
 }  
 }  
  
 for (int i = 0; i < x.GetLength(0); i++)  
 {  
 for (int j = 0; j < x.GetLength(1); j++)  
 {  
 Console.Write($"{x[i, j],4} ");  
 }  
 Console.WriteLine();  
 }  
 }  
}

**Вопросы.**

1. **Модульное программирование** – это подход к программированию, при котором процесс разработки программы разделяется на отдельные подзадачи, такие как функции, процедуры или объекты.

2. Принципы модульного программирования включают:

- Высокая связность (Cohesion): В рамках одного модуля все составляющие тесно связаны с одной функциональностью.

- Низкая связанность (Coupling): Модули связаны друг с другом слабо, что улучшает переиспользуемость и облегчает изменения.

- Инкапсуляция: Модули скрывают свои внутренности (данные и реализацию) от других модулей, общаясь через определенные интерфейсы.

- Модульность: Программа разделена на модули, которые можно разрабатывать и тестировать отдельно друг от друга.

- Независимость: Модули можно компилировать независимо, позволяя минимизировать влияние изменений в одном модуле на другие модули.

3. Характеристики модуля:

- Функциональная независимость: Способность модуля выполнять специфичную задачу независимо от других частей системы.

- Четкие интерфейсы: Ясно определенные способы взаимодействия модуля с другими частями программы.

- Переиспользуемость: Способность использовать модуль в различных программах без изменений или с минимальной адаптацией.

- Легкость подключения: Простота интеграции модуля в систему.

- Легкость тестирования и отладки: Возможность проверки работоспособности модуля отдельно от остальной программы.

4. Свойства, которыми должен обладать модуль:

- Высокая когезия: Все элементы модуля должны быть сгруппированы вокруг общей задачи или функции и взаимозаменяемы в пределах этой задачи.

- Низкая связность: Модуль должен иметь минимальные ссылки на другие модули.

- Абстракция: Модуль должен представлять отдельную функциональность на достаточно высоком уровне абстракции, избегая чрезмерных деталей.

- Независимость: Модуль должен иметь возможность функционировать и тестироваться независимо от остальной системы.

- Простота замены: Модуль должен быть создан таким образом, чтобы его можно было легко заменить другим модулем с аналогичным интерфейсом.

- Прозрачность: Взаимодействие между модулем и использованием его функциональности должно быть понятным для других модулей или разработчиков.