# Упражнения: Прости пресмятания

Задачи за упражнение в клас и за домашно към курса [„Основи на програмирането“ @ СофтУни](https://softuni.bg/courses/programming-basics).

## Празно Visual Studio решение (Blank Solution)

Създайте празно решение (**Blank Solution**) във Visual Studio. Решенията (solutions) във Visual Studio обединяват **група проекти**. Тази възможност е изключително удобна, когато искаме да работим по няколко проекта и бързо да превключваме между тях или искаме да обединим логически няколко взаимосвързани проекта.

В настоящото практическо занимание ще използваме **Blank Solution с няколко проекта** за да организираме решенията на задачите от упражненията – всяка задача в отделен проект и всички проекти в общ solution.

1. Стартирайте Visual Studio.
2. Създайте нов **Blank Solution**: [File]🡪 [New] 🡪 [Project].



1. Изберете от диалоговия прозорец [Templates] 🡪 [Other Project Types] 🡪 [Visual Studio Solutions] 🡪 [**Blank Solution**] и дайте подходящо име на проекта, например “Simple-Calculations”:



Сега имате създаден **празен Visual Studio Solution** (с 0 проекта в него):



Целта на този blank solution e да добавяте в него **по един проект за всяка задача** от упражненията.

## Пресмятане на лице на квадрат

Първата задача от тази тема е следната: да се напише **конзолна програма**, която **въвежда цяло число** „a“ и **пресмята лицето на квадрат** **със страна** „a“. Задачата е тривиално лесна: въвеждате число от конзолата, умножавате го само по себе си и печатате получения резултат на конзолата.

1. Създайте **нов проект** в съществуващото Visual Studio решение. В Solution Explorer кликнете с десен бутон на мишката върху **Solution 'Simple-Calculations'**. Изберете [Add] 🡪 [New Project…]:



1. Ще се отвори диалогов прозорец за избор на тип проект за създаване. Изберете C# конзолно приложение с име “Square-Area”:



Вече имате solution с едно конзолно приложение в него. Остава да напишете кода за решаване на задачата.

1. Отидете в тялото на метода Main(string[] args) и напишете кода от картинката по-долу:



Кодът въвежда цяло число с a = int.Parse(Console.ReadLine()), след това изчислява area = a \* a и накрая печата стойността на променливата area.

1. **Стартирайте** програмата с [Ctrl+F5] и я **тествайте** с различни входни стойности:



1. **Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#0>. Трябва да получите 100 точки (напълно коректно решение):





## От инчове към сантиметри

Да се напише програма, която **чете от конзолата число** (не непременно цяло) и преобразува числото **от инчове в сантиметри**. За целта **умножава инчовете по 2.54** (защото 1 инч = 2.54 сантиметра).

1. Първо създайте **нов C# конзолен проект** в решението “**Simple-Calculations**”. Кликнете с мишката върху решението в Solution Explorer и изберете [Add] 🡪 [New Project…]:



Изберете [Visual C#] 🡪 [Windows] 🡪 [Console Application] и задайте име “Inches-to-Centimeters”:



1. **Напишете кода** на програмата. Може да си помогнете с примерния код от картинката:



1. **Стартирайте програмата**, както обикновено с [Ctrl+F5]:



Изненада! Како става? Програмата не работи правилно… Всъщност това не е ли предходната програма?

Във Visual Studio **текущият активен проект** в един solution е маркиран в получерно и може да се сменя:



1. За да включите режим на **автоматично преминаване към текущия проект**, кликнете върху главния solution с десния бутон на мишката и изберете **[Set StartUp Projects…]**:



Ще се появи диалогов прозорец, от който трябва да се избере **[Startup Project]** 🡪 [**Current selection**]:



1. Сега отново **стартирайте програмата**, както обикновено с [Ctrl+F5]. Този път ще се стартира текущата отворена програма, която преобразува инчове в сантиметри. Изглежда работи коректно:



1. Сега **превключете към преходната програма** (лице на квадрат). Това става с двоен клик на мишката върху файла Program.cs от предходния проект “Square-Area” в панела [Solution Explorer] на Visual Studio:



1. Натиснете пак **[Ctrl+F5]**. Този път трябва да се стартира другият проект:



1. Превключете обратно към проекта “Inches-to-Centimeters” и го стартирайте с **[Ctrl+F5]**:



**Превключването между проектите** е много лесно, нали? Просто избираме файла със сорс кода на програмата, кликваме го два пъти с мишката и при стартиране тръгва програмата от този файл.

1. Тествайте с **дробни числа**, например с 2.5:



**Внимание:** в зависимост от регионалните настройки на операционната система, е възможно вместо **десетична точка** (US настройки) да се използва **десетична запетая** (BG настройки). Ако програмата очаква десетична точка и бъде въведено число с десетична запетая или на обратно (бъде въведена десетична точка когато се очаква десетична запетая), ще се получи следната грешка:



Препоръчително е **да промените настройките на компютъра си**, така че да се използва **десетична точка**:





1. Вече е време за **тестване в judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#1>. Решението би трябвало да бъде прието като напълно коректно:



## Поздрав по име

Да се напише програма, която **чете от конзолата име на човек** и отпечатва “Hello, <name>!”, където <name> е въведеното преди това име.

1. Първо създайте **нов C# конзолен проект** с име “Greeting” в решението “Simple-Calculations”:



1. **Напишете кода** на програмата. Ако се затруднявате, може да ползвате примерния код по-долу:



1. **Стартирайте** програмата с **[Ctrl+F5]** и я тествайте:



1. Тествайте в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#2>. Преди да пратите решението сложете коментар на първия ред, който печата “Enter your name”.

## Съединяване на текст и числа

Напишете C# програма, която прочита от конзолата име, фамилия, възраст и град и печата съобщение от следния вид: “You are <firstName> <lastName>, a <age>-years old person from <town>”.

1. Добавете към текущото Visual Studio решение още един **конзолен C# проект** с име “Concatenate-Data”.
2. **Напишете кода**, който чете входните данни от конзолата:



1. **Допишете код**, който отпечатва описаното в условието на задачата съобщение.



На горната картинка кодът е нарочно даден размазан, за да помислите как да си го напишете сами.

1. **Тествайте** решението локално с [Ctrl+F5] и въвеждане на примерни данни.
2. Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#3>.

## Лице на трапец

Напишете програма, която чете от конзолата три числа b1, b2 и h и **пресмята лицето на трапец** с основи b1 и b2 и височина h. **Формулата за лице на трапец** е (b1 + b2) \* h / 2.

На фигурата по-долу е показан трапец със страни 8 и 13 и височина 7. Той има лице (8 + 13) \* 7 / 2 = 73.5.



1. Добавете към текущото Visual Studio решение още един **конзолен C# проект** с име “Trapezoid-Area”.
2. **Напишете кода**, който чете входните данни от конзолата, пресмята лицето на трапеца и го отпечатва:



Кодът на картинката е нарочно размазан, за да си го доизмислите и допишете сами.

1. **Тествайте** решението локално с [Ctrl+F5] и въвеждане на примерни данни.
2. Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#4>.

## Периметър и лице на кръг

Напишете програма, която чете от конзолата **число** r и пресмята и отпечатва **лицето** и **периметъра на кръг** / окръжност с радиус r.

|  |  |
| --- | --- |
| **вход** | **изход** |
| 3 | Area = 28.2743338823081  Perimeter = 18.8495559215388 |
| 4.5 | Area = 63.6172512351933  Perimeter = 28.2743338823081 |

За изчисленията можете да използвате следните формули:

* area = Math.PI \* r \* r
* perimeter = 2 \* Math.PI \* r

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#5>.

## Лице на правоъгълник в равнината

**Правоъгълник** е зададен с **координатите** на два от своите срещуположни ъгъла (x1, y1) – (x2, y2). Да се пресметнат **площта** и **периметъра** му. **Входът** се чете от конзолата. Числата x1, y1, x2 и y2 са дадени по едно наред. **Изходът** се извежда на конзолата и трябва да съдържа два реда с по една число на всеки от тях – лицето и периметъра.

|  |  |
| --- | --- |
| **вход** | **изход** |
| 60  20  10  50 | 1500  160 |
| 30  40  70  -10 | 2000  180 |
| 600.25  500.75  100.50  -200.5 | 350449.6875  2402 |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#6>.

## Лице на триъгълник

Напишете програма, която чете от конзолата **страна** и **височина** на **триъгълник** и пресмята неговото лице. Използвайте **формулата** за лице на триъгълник: area = a \* h / 2. Закръглете резултата до **2 знака след десетичната точка** използвайки [**Math.Round(area, 2)**](https://msdn.microsoft.com/en-us/library/75ks3aby(v=vs.110).aspx).

|  |  |
| --- | --- |
| **вход** | **изход** |
| 20  30 | Triangle area = 300 |
| 15  35 | Triangle area = 262.5 |
| 7.75  8.45 | Triangle area = 32.74 |
| 1.23456  4.56789 | Triangle area = 2.82 |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#7>.

## Конзолен конвертор: от градуси °C към градуси °F

Напишете програма, която чете **градуси по скалата на Целзий** (°C) и ги преобразува до **градуси по скалата на Фаренхайт** (°F). Потърсете в Интернет подходяща [формула](http://bfy.tw/3rGh), с която да извършите изчисленията. Закръглете резултата до **2 знака след десетичната точка**. Примери:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 25 | 77 | 0 | 32 | -5.5 | 22.1 | 32.3 | 90.14 |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#8>.

## Конзолен конвертор: от радиани в градуси

Напишете програма, която чете **ъгъл в** [**радиани**](https://bg.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B4%D0%B8%D0%B0%D0%BD) (rad) и го преобразува в [**градуси**](https://bg.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%B4%D1%83%D1%81_(%D1%8A%D0%B3%D1%8A%D0%BB)) (deg). Потърсете в Интернет подходяща формула. Числото **π** в C# програми е достъпно чрез Math.PI. Закръглете резултата до най-близкото цяло число използвайки Math.Round(). Примери:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 3.1416 | 180 | 6.2832 | 360 | 0.7854 | 45 | 0.5236 | 30 |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#9>.

## Конзолен конвертор: USD към BGN

Напишете програма за **конвертиране на щатски долари** (USD) **в български лева** (BGN). **Закръглете** резултата до **2 цифри** след десетичната запетая. Използвайте фиксиран **курс** между долар и лев: **1 USD** = **1.79549 BGN**.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 20 | 35.91 BGN | 100 | 179.55 BGN | 12.5 | 22.44 BGN |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#10>.

## \* Конзолен междувалутен конвертор

Напишете програма за **конвертиране на парична сума от една валута в друга**. Трябва да се поддържат следните валути: **BGN**, **USD**, **EUR**, **GBP**. Използвайте следните фиксирани валутни курсове:

|  |  |  |  |
| --- | --- | --- | --- |
| **Курс** | **USD** | **EUR** | **GBP** |
| **1 BGN** | 1.79549 | 1.95583 | 2.53405 |

**Входът** e **сума за конвертиране** + **входна валута** + **изходна валута**. **Изходът** е едно число – преобразуваната сума по посочените по-горе курсове, закръглен до **2 цифри** след десетичната точка. Примери:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 20  USD  BGN | 35.91 BGN | 100  BGN  EUR | 51.13 EUR | 12.35  EUR  GBP | 9.53 GBP | 150.35  USD  EUR | 138.02 EUR |

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#11>.

## \*\* Пресмятане с дати: 1000 дни на Земята

Напишете програма, която въвежда **рождена дата** във формат “dd-MM-yyyy” и пресмята датата, на която се навършват **1000 дни** от тази рождена дата и я отпечатва в същия формат.

|  |  |
| --- | --- |
| **вход** | **изход** |
| 25-02-1995 | 20-11-1997 |
| 07-11-2003 | 02-08-2006 |
| 30-12-2002 | 24-09-2005 |
| 01-01-2012 | 26-09-2014 |
| 14-06-1980 | 10-03-1983 |

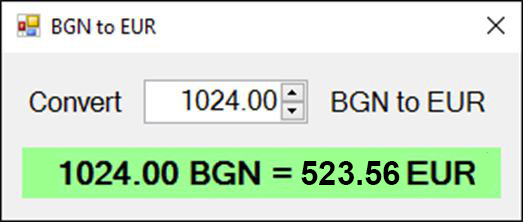
**\* Подсказки**: потърсете информация за типа DateTime в C# и по-конкретно разгледайте методите ParseExact(str, format), AddDays(count) и ToString(format). С тяхна помощ може да решите задачата, без да е необходимо да изчислявате дни, месеци и високосни години.

Тествайте решението си в **judge системата**: <https://judge.softuni.bg/Contests/Practice/Index/151#12>. Не печатайте нищо допълнително на конзолата освен изискваната дата!

# Упражнения: Графични и Web приложения

## Графично приложение: конвертор от BGN към EUR

Създайте **графично приложение** (GUI application), което пресмята стойността в **евро** (EUR) на парична сума, зададена в **лева** (BGN). При промяна на стойността в лева, равностойността в евро трябва да се преизчислява автоматично. Използвайте курс лева / евро: 1.95583.



1. Добавете към текущото Visual Studio решение още един проект. Този път създайте **Windows Forms** приложение със C# с име “BGN-to-EUR-Converter”:



1. Подредете следните UI контролите във формата:

* NumericUpDown с име numericUpDownAmount – ще въвежда сумата за конвертиране
* Label с име labelResult – ще показва резултата след конвертиране
* Още два Label компонента, служещи единствено за статично изобразяване на текст

Графичният редактор за потребителски интерфейс може да изглежда по подобен начин:



1. Задайте настройки на формата и на отделните контроли:

FormConverter



* Text = "BGN to EUR"
* Font.Size = 12
* MaximizeBox = False
* MinimizeBox = False
* FormBorderStyle = FixedSingle

numericUpDownAmount



* Value = 1
* Minimum = 0
* Maximum = 10000000
* TextAlign = Right
* DecimalPlaces = 2

labelResult

****

* AutoSize = False
* BackColor = PaleGreen
* TextAlign = MiddleCenter
* Font.Size = 14
* Font.Bold = True

1. Дефинирайте **обработчици на събития** по контролите:



Хванете следните събития:

* FormConverter.Load (кликнете върху формата с мишката 2 пъти)
* numericUpDownAmount.ValueChanged (кликнете върху NumericUpDown контролата 2 пъти)
* numericUpDownAmount.KeyUp (изберете **Events** от таблото **Properties** и кликнете 2 пъти върху **KeyUp**)

Събитието Form.Load се изпълнява при стартиране на програмата, преди да се появи прозореца на приложението. Събитието NumericUpDown.ValueChanged се изпълнява при промяна на стойността в полето за въвеждане на число. Събитието NumericUpDown.KeyUp се изпълнява след натискане на клавиш в полето за въвеждане на число. При всяко от тези събития ще преизчисляваме резултата.

За **хващане на събитие** ползвайте иконката със събитията в **Properties** прозореца във Visual Studio:



Сложете следния **C#** **код** за обработка на събитията:

|  |
| --- |
| private void FormConverter\_Load(object sender, EventArgs e)  {  ConvertCurrency();  }  private void numericUpDownAmount\_ValueChanged(object sender, EventArgs e)  {  ConvertCurrency();  }  private void numericUpDownAmount\_KeyUp(object sender, KeyEventArgs e)  {  ConvertCurrency();  } |

Всички прихванати събития извикват метода ConvertCurrency(), който конвертира зададената сума от лева в евро и показва резултата в зелената кутийка.

1. **Напишете кода** (програмната логика) за конвертиране от лева към евро:

|  |
| --- |
| private void ConvertCurrency()  {  var amountBGN = this.numericUpDownAmount.Value;  var amountEUR = amountBGN \* 1.95583m;  this.labelResult.Text =  amountBGN + " BGN = " +  Math.Round(amountEUR, 2) + " EUR";  } |

1. **Стартирайте** проектa с [Ctrl+F5] и тествайте дали работи правилно.

## \*\*\* Графично приложение: хвани бутона!

Създайте забавно графично приложение „**хвани бутона**“: една **форма** съдържа един **бутон**. При преместване на курсора на мишката върху бутона той се премества на случайна позиция. Така се създава усещане, че „**бутонът бяга от мишката** и е трудно да се хване“. При „хващане“ на бутона се извежда съобщение-поздрав.

**\* Подсказка**: напишете обработчик за събитието Button.MouseEnter и премествайте бутона на случайна позиция. Използвайте генератор за случайни числа Random. Позицията на бутона се задава от свойството Location. За да бъде новата позиция на бутона в рамките на формата, можете да направите изчисления спрямо размера на формата, достъпен от свойството ClientSize. Можете да ползвате следния код за ориентир:

|  |
| --- |
| private void **buttonCatchMe**\_**MouseEnter**(object sender, EventArgs e)  {  Random rand = new Random();  var maxWidth = this.ClientSize.Width - buttonCatchMe.ClientSize.Width;  var maxHeight = this.ClientSize.Height - buttonCatchMe.ClientSize.Height;  this.buttonCatchMe.Location = new Point(  rand.Next(maxWidth), rand.Next(maxHeight));  } |