



Пловдивски университет  
„Паисий Хилендарски“



---

**ФАКУЛТЕТ ПО МАТЕМАТИКА И ИНФОРМАТИКА**

**КАТЕДРА „ОБУЧЕНИЕ ПО МАТЕМАТИКА, ИНФОРМАТИКА И  
ИНФОРМАЦИОННИ ТЕХНОЛОГИИ“**

# Дипломна работа

Тема: Въведение в програмирането  
със C# в общообразователната  
подготовка по информатика

**Дипломант:** Дейвид Младенов      **Научен ръководител:**

**Факултетен номер:** 1601651011      проф. д-р Коста Гъров

**Специалност:** ИТМОМ

# Съдържание

I. Увод .....	3
II. Учебна програма по информатика за VIII клас (общообразователна подготовка)....	8
III. Учебно помагало „Въведение в програмирането със C#“ .....	21
1. Задачите в обучението по програмиране.....	21
2. Бройни системи.....	21
3. Интегрирана среда за разработка (IDE).....	24
4. Създаване на решение (Solution).....	26
5. Първа програма .....	29
6. Примитивни типове данни и променливи .....	35
7. Оператори .....	37
8. Вход и изход от конзолата .....	39
9. Условни конструкции.....	42
10. Цикли.....	46
11. Масиви.....	50
12. Методи.....	52
13. Основни методи при работа със символни низове .....	55
14. Обработка на изключения .....	56
15. Създаване и използване на обекти .....	58
16. Приложения с графичен потребителски интерфейс .....	65
IV. Контрол и оценяване на усвоените знания .....	74
V. Заключение.....	84
VI. Използвана литература .....	85
VII. Приложение с решения на задачите от дипломната работа .....	86
1. Решения на задачите от разработените теми .....	86
2. Решения на задачите от контрол и оценяване.....	107

# І. Увод

България може да се похвали с факта, че е една от първите държави в света, които въвеждат изучаването на информатика в средните училища. През 60-те години в много окръжни градове на страната се създават т.н. математически паралелки. В тях учениците изучават предметите “Програмиране” и “Числени методи”. В часовете по “Програмиране” са застъпени темите: Бройни системи, Математически и логически основи на ЕИМ (Електронно изчислителни машини), Алгоритми и начини за изразяването им, Програмиране на АЛГОЛ и ФОРТРАН. По “Числени методи” се разглеждат: Матрици и детерминанти, Приближено решаване на уравнения и системи линейни уравнения, Приближаване на функции, Симплекс-метод, Транспортна задача. Занятията се водят теоретично, като практическа работа се извършва само с електронни калкулатори. Провеждат се и наблюдения върху работата на ЕИМ в големи изчислителни центрове [7].

В началото на 70-те години се създават математическите гимназии, в които обучението по информатика се провежда по същия начин, както в математическите паралелки. През 1976 г. в Математическата гимназия в Пловдив се открива първият Учебен електронно-изчислителен център в страната, снабден с ЕИМ ИЗОТ 310. По-късно такива центрове се откриват и в София, Русе и Варна. Те са оборудвани с големи ЕИМ от серията ЕС. Наличието на УЕИЦ позволява съществено да се увеличи делът на практическата дейност на учениците.

В началото на 80-те години се прави опит знания по информатика да се преподават в часовете по математика. През 1981 г. в учебника по алгебра за 9. клас около половината от съдържанието е отделено на информатични знания. Този учебник съдържа раздели за бройни системи, алгоритми, приближени стойности, за електронните калкулатори “ЕЛКА” и за автоматични сметачни машини. Подобен подход – да се интегрират елементи от информатиката в други учебни дисциплини, е възприет в системата от училища на Проблемната група по образование (ПГО), където се осъществява експериментално обучение под ръководството на академик Благовест Сендов.

През 1983 г. се осъществява поредната реформа в българското средно образование. Според нея в първата степен на обучение (1.-10.) клас се осъществява общото образование на учениците, а във втората степен (11.-12.) клас се овладява професия в т. н. Учебно-професионални комплекси (УПК). За учениците от математическите гимназии и някои техникуми е въведена подготовка за професията “Оператор-програмист на ЕИМ”. Изучават се следните общотехнически и специални предмети: Информатика, Приложна математика, Операционни системи, Програмиране и алгоритмични езици, Електронно-изчислителна техника, Икономика на производството и стопанското управление, Електротехника и електроника. Независимо от редица слабости в някои от учебните програми, въвеждането на тази професия е голяма стъпка напред при изучаване на знания по информатика в средното училище.

Междувременно в Института по техническа кибернетика и роботика (ИТКР) се създава първият български микрокомпютър “ИМКО-2”. През март 1983 г. в МГ “Акад. К. Попов”, Пловдив се открива първият в страната компютърен кабинет, оборудван с 8-битови компютри “Правец 82”. Следва период на бързо насищане на българските училища с микрокомпютри от този вид.

Така в средата на 80-те години информатиката навлиза под различни форми в средното училище, но е неравномерно застъпена в различните класове, не обхваща най-съществените за средното образование въпроси, недостатъчна е по обем и с незначителни изключения преподаването е с незадоволително качество. Аналогично е положението с материалната база – използва се предимно амортизирана изчислителна техника и все още малък брой микрокомпютри.

През 1986 г. МНП взима изключително важното решение да въведе отделен учебен предмет “Информатика” в 10. и 11. клас на средното общообразователно училище (СОУ). Учебната програма в значителна степен е преходна. Тя е съобразена със съществуващото положение в страната: широко разпространение на микрокомпютъра “Правец 82” и езика Бейсик.

За пръв път в нашата образование са разработени 3 различни учебника по една учебна дисциплина – Информатика, ориентирани към ученици от различните видове училища. Авторски колектив под ръководството на професор П. Бърнев написва учебници, предназначени за т. н. “елитни” училища – езикови и математически гимназии – [3] и [4]. А. Ангелов, К. Гъргов и О. Гавраилов създават учебници, предназначен за масовото средно училище – [1] и [2]. Колектив под ръководството на проф. Л. Даковски написва учебник, ориентиран към учениците от техникумите и СПТУ.

С бързи темпове продължава оборудването на училищата с микрокомпютри “Правец”. По данни на МНП за периода 1984 – 1989 г. българските училища са оборудвани с около 15 000 компютри.

През 1990 г. се извършва промяна в учебния план на СОУ, при която учебният предмет Информатика се изучава в 9. и 10. клас. Съгласно учебната програма в 9. клас, основните понятия и процеси се демонстрират с програми на езика ИНФО. Този алгоритмичен език е подобен на езика за програмиране Паскал, като командите се изписват на български език. В 10 клас е предвидено изучаването на различни приложения на компютрите в практиката.

През 1994 г. настъпва поредната промяна в учебния план на СОУ, при която Информатика се изучава само в 11. клас с 2 часа седмично. Преподаването се извършва по учебното помагало “Информатика” с автори О. Гавраилов и К. Гъргов [5]. За училища, оборудвани с 16 битови компютри в 9. и 10. клас е въведен нов учебен предмет “Информационни технологии” (ИТ). ИТ, базирани на компютърни системи, формират един от най-бързо развиващите се технологични клонове в съвременния стопански и обществен живот. Добилият популярност термин “информационни технологии” се използва в практиката за отбелязване на много широк спектър от конкретни продукти, технологии, технологични процеси и дейности, както в областта на производството на компютърни, комуникационни и офис-системи, така и в областта на създаването и експлоатацията на софтуерни продукти. В Указанието на Министерството на науката и образованието (МНО) за организиране на обучението по Информационни технологии в българските училища [9], се посочва следното работно определение за ИТ:

Технологии, свързани с разработването и/или използването на програмни продукти и системи, предназначени да автоматизират дейностите по реализиране на основните информационни процеси (събиране, съхраняване, преработка и разпространение на информация) чрез използване на компютри, ще наричаме информационни.

Целта на това определение е прагматично да стесни обхвата, очертае рамките и уточни значението на понятието “информационни технологии” за нуждите на СОУ, като по този начин формира критерии, според които да се определя съдържанието на обучението по ИТ. По учебния предмет ИТ са разработени и утвърдени от МОН следните осем учебни програми:

- Операционни системи с текстов интерфейс;
- Текстообработка;
- Електронни таблици;
- Базы от данни;
- Компютърна графика (проектиране);
- Информационни технологии за математически изследвания;
- Графичен потребителски интерфейс MS Windows;
- “Интернет за начинаещи” и “Интернет за напреднали”.

По време на обучението се изучават три или четири от посочените модули, в зависимост от вида на техниката, профила на училището и желанията на учениците.

При така очертаната картина на училищната информатика в края на XX век можем да посочим някои проблеми, свързани с нея:

- Информатиката като нов учебен предмет трябва да се “вмести” в учебния план сред вече утвърдени учебни дисциплини. Това естествено е наложило намаляване на хорариума на други дисциплини и е породило негативно отношение на някои специалисти от МОН към новия предмет. Информатиката е започнала да променя своето място от едни класове в други и да намалява обема си в учебния план.

- Опитът “да се размиват” знанията по информатика в часовете по математика едва ли е можел да се приеме като успешен. Достатъчно е да се спомене фактът, че не всички преподаватели по математика имат необходимата квалификация по информатика, за да стане ясно, че реални положителни резултати от този подход не могат да се очакват.

- Наличната компютърна техника в страната по това време като количество и качество не създава предпоставки за всеобщо и единно обучение по информатика.

- Подготовката на учители по информатика не е била на необходимото равнище. Бързото развитие на ИТ поставя въпроса и за преквалификацията им през определен период от време.

В началото на XXI век у нас започва нова реформа в образователната система. Тя е била породена както от настъпилите обществено-политически промени в България, така и от технологичните промени в световен мащаб. Съвременното обществено развитие, преходът от индустриално към постиндустриално общество в световен мащаб налага глобализация на икономиката, висока мобилност на работната сила и изисква нов тип компетентности на личността в социалната и професионалната сфера. Тези процеси рефлектират особено силно в сферата на образованието. Образователната политика на държавата отчита както социално-икономическите характеристики на съвременното общество, така и перспективите за бъдещото му развитие. В този смисъл днес се налага

преосмисляне на способността на българското училище да отговори адекватно на новите предизвикателства.

Наши и чужди изследвания от последните години показват, че учебното съдържание по различните предмети в България:

- е с подчертано академичен характер за голяма част от тях;
- е голямо по обем;
- в недостатъчна степен отчита възрастовите особености на децата и учениците;
- е подчинено предимно на изискването за запаметяване и възпроизвеждане на конкретни знания;
- е затворено в отделните предмети, без да позволява достатъчно продуктивни връзки между тях;
- в недостатъчна степен се опира на личния опит на ученика;
- като цяло не е ориентирано към формиране на готовност за реализация в съвременното общество.

През последните години световните тенденции, свързани със съдържанието и организацията на обучението, могат да се обединят в четири групи:

- 1) Ориентация към разбиране и осмисляне на знанието.
- 2) Ограничаване ролята на репродуктивното знание.
- 3) Стимулиране на творческата активност на учениците.
- 4) Утвърждаване на единни държавни изисквания.

Необходимостта от държавни изисквания (стандарти) за учебно съдържание е призната единодушно. Това се потвърждава и от факта, че в повечето европейски страни през последните години разработването на такива стандарти се реализира като приоритетна задача. По този начин образователните институции се ангажират с конкретна отговорност за качеството на образованието на национално ниво.

През 1999 г. със заповед на Министъра на образованието и науката са сформирани работни групи за изготвяне на държавни образователни изисквания по съответните учебни предмети. Проектите за стандартите са обсъждани от учители и преподаватели във ВУЗ. След одобряването им, са публикувани на 18.05.2000 г. в Държавен вестник и вече определят насоките на средното образование през периода 2000-2008 година [6].

Централно място в стандартите заемат т. н. ядра на учебно съдържание - до 5-6 на брой за даден предмет. В стандартите за 9.-12. клас, където попадат учебните предмети Информатиката и ИТ за всяко ядро се определят по две равнища - първо и второ.

Първо равнище: Стандартите за това равнище определят общообразователния минимум по предмета. Това са знанията, уменията и отношенията, които учениците могат и трябва да усвоят в рамките на задължителната подготовка. За предмета

Информатика тя е 2 часа седмично в 9. клас, а за ИТ – по 1 час седмично в 9. и 10. клас. Тези стандарти трябва да са постижими за около 80% от учениците.

Второ равнище: Стандартите за това равнище определят съдържанието на профилираната подготовка по съответния предмет. Това са знанията, уменията и отношенията, които учениците могат и трябва да усвоят в допълнителни учебни часове – около 500 в 9.-12. клас.

По предмета Информатика са определени следните ядра на учебно съдържание: Информация и формални модели, Компютърни системи, Операционни системи, Алгоритми и структури от данни, Програмиране.

По Информационни технологии ядрата са: Решаване на проблеми с ИТ, Комуникиране, Контрол и управление на обекти, Моделиране, Интегриране на дейности и продукти в ИТ.

На базата на държавните образователни изисквания бързо са съставени учебни програми по учебните предмети Информатика и Информационни технологии [10]. През 2001 г. са проведени конкурси за написване на учебници по двата учебни предмета. Пълен списък на одобрените учебници за българските училища е даден в [8].

Поради бързото развитие на информационните и комуникационни технологии и масовото им приложение в практиката от учебната 2006-2007 година ИТ започват да се изучават в задължителната подготовка в 5.-7. клас, а в избираемата подготовка от 1.-4. клас. Днес можем да констатираме, че Информатиката и Информационните технологии са утвърдени учебни предмети в българското училище, които се радват на голям интерес от страна на учениците. Изучаването на информатика в училище дава стабилна основа на младите ученици за развитие в областта.

Обект на тази разработка ще бъде съставянето на учебно помагало за упражнение съгласно „Учебна програма по информатика за VIII клас“ (Общообразователна подготовка) и далеч по-модерни технологии за програмиране от споменатите по-горе в лицето на езика C# и .NET Core 3.1 платформата.

C# е съвременен обектно-ориентиран език за програмиране разработен от Microsoft, който се появява пред света през 2000-та година. В днешно време C# е един от най-популярните езици за програмиране и се използва от милиони разработчици по цял свят. Като език от високо ниво C#, се слави с лесния си синтаксис и се препоръчва за начинаещи, които имат интерес в сферата на програмирането и именно за това съм избрал точно него за примерите в тази разработка.

## II. Учебна програма по информатика за VIII клас (общообразователна подготовка)

### Очаквани резултати от обучението в края на класа

Област на компетентност	Знания, умения и отношения В резултат на обучението ученикът:
<b>Информатика</b>	Описва предмета и ролята на информатиката за моделиране
	Познава представянето на информация във вид на данни
	Посочва примери на обекти и явления, при които е практически приложимо използването на средствата на обектно-ориентираното моделиране
<b>Обектно-ориентирано програмиране</b>	Обяснява основните етапи при създаване и изпълнение на компютърна програма
	Прилага обектно-ориентиран подход при създаване на несложна компютърна програма
	Описва основни начини за създаване, изпълнение и тестване на програмен проект в интегрирана среда за разработка с използване на визуални графични средства
	Използва библиотеки от готови компоненти
	Спазва добър стил на програмиране
<b>Графичен потребителски интерфейс</b>	Разбира и използва основни компоненти на среда за визуално програмиране при разработка на софтуер
	Проектира графичен потребителски интерфейс с визуални средства
	Избира подходяща графична компонента в съответствие с необходимата функционалност на графичния интерфейс
	Умее да настройва свойствата на графичните компоненти
	Програмира подразбиращи се събития за основни компоненти от графичния интерфейс
	Разбира същността на „тип данни“
	Разграничава различни типове данни



<b>Алгоритми и структури от данни</b>	Определя подходящ тип данни за определена задача
	Разбира същността на алгоритмите и начини за описанието им
	Прилага основни управляващи конструкции
	Структурира данни в едномерен масив
	Прилага основни алгоритми за намиране на сума, минимален/максимален елемент и средно аритметично
	Чете и записва данни в текстов файл
<b>Софтуерни приложения</b>	Използва визуално програмиране за решаване на несложни задачи
	Създава програми за графично изобразяване на геометрични обекти със стандартни средства в езика
	Създава програмни приложения с мултимедийни компоненти
	Представя аргументирано разработено софтуерно приложение пред публика

### Учебно съдържание

Теми	Компетентности като очаквани резултати от обучението	Нови понятия
<b>1. Основи на информатиката</b>		
1.1. Числата и техните представяния	<ul style="list-style-type: none"> <li>Разширява и обобщава знанията, свързани с числата и техните представяния: <ul style="list-style-type: none"> <li>непозиционни бройни системи;</li> <li>същност на позиционните бройни системи;</li> <li>формат на числата в десетична, двоична и шестнадесетична бройна система.</li> </ul> </li> <li>Превръща числа от десетична в двоична бройна система и обратно.</li> </ul>	<ul style="list-style-type: none"> <li>непозиционни бройни системи</li> <li>позиционни бройни системи</li> <li>експоненциален формат и неговото предназначение</li> <li>двоична бройна система</li> <li>шестнадесетична бройна система</li> </ul>

	<ul style="list-style-type: none"> <li>• Извършва събиране, изваждане и умножение на две числа в двоична бройна система.</li> <li>• Дава примери за използване на двоична и шестнадесетична бройна система.</li> </ul>	
1.2. Информационни дейности и процеси	<ul style="list-style-type: none"> <li>• Обяснява предмета на информатиката и ролята ѝ в съвременното общество.</li> <li>• Изброява и описва основните информационни дейности събиране, съхраняване, преработка и разпространение и общата схема на информационните потоци.</li> <li>• Описва понятието информационен процес и дава примери на информационни процеси, свързани с решаване на житейски задачи.</li> <li>• Различава понятията информация и данни.</li> <li>• Обяснява и илюстрира с примери връзката между информация и данни.</li> <li>• Обяснява и илюстрира с примери същността на дискретното представяне на информацията за трансформирането ѝ в данни.</li> </ul>	<ul style="list-style-type: none"> <li>• основни информационни дейности</li> <li>• информация, данни, дискретно представяне на информацията</li> </ul>
1.3. Алгоритми и езици за програмиране	<ul style="list-style-type: none"> <li>• Дефинира понятието алгоритъм и описва основните му характеристики (резултатност, крайност, детерминираност, масовост).</li> <li>• Описва и проследява несложни, линейни и разклонени алгоритми с различни средства.</li> </ul>	<ul style="list-style-type: none"> <li>• алгоритъм</li> <li>• език за програмиране</li> <li>• транслятор</li> </ul>

	<ul style="list-style-type: none"> <li>• Описва същността, структурата и разновидностите на цикличните алгоритмични конструкции.</li> <li>• Обяснява същността и функционалното предназначение на език за програмиране.</li> <li>• Обяснява същността и предназначението на транслятор (интерпретатор, компилатор).</li> <li>• Представя исторически факти, свързани със създаването и развитието на съвременните езици и среди за програмиране.</li> </ul>	
<b>2. Среда за визуално програмиране</b>		
2.1. Интегрирана среда за визуално програмиране	<ul style="list-style-type: none"> <li>• Отваря проект в интегрирана среда за визуално програмиране.</li> <li>• Редактира дизайна на графичния потребителски интерфейс на приложение в интегрирана среда за програмиране.</li> <li>• Запазва проект на приложение чрез средствата на интегрирана среда за програмиране.</li> <li>• Стартира приложение с графичен потребителски интерфейс чрез средствата на интегрирана среда за програмиране.</li> <li>• Разпознава основни компоненти на интегрирана среда за програмиране – графичен и текстов редактор, панел с контроли, панел за свойства на обект, панел за съобщения, панел за преглед</li> </ul>	<ul style="list-style-type: none"> <li>• интегрирана среда за програмиране</li> <li>• свързващ редактор (linker)</li> <li>• програма за откриване и отстраняване на грешки (debugger)</li> <li>• редактор за проектиране на дизайн на графичен потребителски интерфейс</li> </ul>

	на структурата на приложението.	
2.2. Основни етапи на създаване и изпълнение на компютърна програма	<ul style="list-style-type: none"> <li>• Анализира задача с несложен математически модел.</li> <li>• Създава математически модел за решаване на несложна задача.</li> <li>• Съпоставя математически модел с програмно решение на даден проблем.</li> <li>• Открива основните компоненти на математически модел в демонстрирано програмно решение на даден проблем.</li> <li>• Стартира чрез средствата на интегрирана среда предварително подготвена компютърна програма с графичен потребителски интерфейс.</li> <li>• Тества предварително подготвен несложен проект.</li> <li>• Разпознава видовете грешки при програмиране.</li> <li>• Разчита и прави предположение за естеството на синтактична грешка в даден проект.</li> <li>• Открива и прави предположение за причината за логическа грешка в дадено приложение.</li> <li>• Открива и прави предположение за причината за грешка по време на изпълнение на приложението.</li> </ul>	<ul style="list-style-type: none"> <li>• синтактични грешки в компютърна програма</li> <li>• логически грешки в компютърна програма</li> <li>• грешки по време на изпълнение на програмата</li> </ul>
2.3. Проектиране на графичен потребителски интерфейс	<ul style="list-style-type: none"> <li>• Знае предназначението на основни контейнери и контроли – форма, етикет,</li> </ul>	<ul style="list-style-type: none"> <li>• интерфейсен компонент (контрола)</li> </ul>

	<p>текстово поле, бутон, диалогова кутия.</p> <ul style="list-style-type: none"> <li>Разпознава основни свойства на графични обекти-контроли – име, състояние, етикет, фон, настройка на шрифт и др.</li> <li>Проектира несложна форма, съдържаща етикет, текстово поле, бутон.</li> <li>Настройва основни свойства на форма, етикет, текстово поле и бутон.</li> <li>Именува обекти-контроли съгласно общоприета конвенция.</li> <li>Задава функционалност на бутон, свързана с извеждането на статично съобщение в диалогова кутия.</li> </ul>	<ul style="list-style-type: none"> <li>контейнер на контроли</li> <li>свойство на обект</li> <li>метод на обект</li> </ul>
<b>3. Програмиране</b>		
<b>3.1. Основни типове данни</b>		
3.1.1. Тип низ	<ul style="list-style-type: none"> <li>Познава правила за именуване на константи и променливи.</li> <li>Декларира, описва и инициализира променливи и константи от тип низ.</li> <li>Присвоява стойност на променлива от тип низ.</li> <li>Въвежда и извежда данни от тип низ в/от текстово поле.</li> <li>Извежда данни от тип низ в/от етикет.</li> <li>Извършва конкатенация на низове.</li> <li>Използва стандартни методи на интерфейсни компоненти за форматиране на текст.</li> </ul>	<ul style="list-style-type: none"> <li>символ</li> <li>низ</li> <li>множество на допустимите данни</li> <li>множество на допустимите операции</li> <li>име, тип стойност на променлива</li> <li>име, тип и стойност на константа</li> <li>присвояване на стойност</li> <li>конкатенация</li> </ul>

<p>3.1.2. Целочислени типове данни</p>	<ul style="list-style-type: none"> <li>• Декларира, описва и инициализира променливи и константи от целочислен тип данни.</li> <li>• Използва вградени функции за преобразуване на низ в цяло число и обратното.</li> <li>• Въвежда и извежда данни от целочислен тип.</li> <li>• Използва различни целочислени типове данни.</li> <li>• Познава целочислените аритметични операции и техния приоритет.</li> <li>• Конструира аритметични изрази, съдържащи само целочислени данни, спазвайки синтаксиса и семантиката на конкретния език за програмиране.</li> <li>• Прилага и анализира резултатите от операциите – събиране, изваждане, умножение, деление, цяла част и остатък от целочислено деление.</li> <li>• Реализира модел за решаване на задачи, базиран на целочислени типове данни.</li> </ul>	<ul style="list-style-type: none"> <li>• целочислен тип данни</li> <li>• конвенция за именуване на константи и променливи</li> </ul>
<p>3.1.3. Реални типове данни</p>	<ul style="list-style-type: none"> <li>• Декларира, описва и инициализира променливи и константи от реален тип.</li> <li>• Използва вградени функции за преобразуване на низ в реално число и обратното.</li> <li>• Въвежда и извежда данни от реален тип данни.</li> <li>• Използва различни реални типове данни.</li> <li>• Познава приоритетите на аритметичните операции при реални типове данни.</li> </ul>	<ul style="list-style-type: none"> <li>• реален тип данни</li> </ul>

	<ul style="list-style-type: none"> <li>• Конструира аритметични изрази, съдържащ реални типове данни, спазвайки синтаксиса и семантиката на конкретния език за програмиране.</li> <li>• Прилага и анализира резултатите от операциите – събиране, изваждане, умножение, деление.</li> <li>• Реализира модел за решаване на задачи, базиран на реални типове данни.</li> </ul>	
<p>3.1.4.</p> <p>Аритметични изрази и вградени математически функции. Приоритет на операциите</p>	<ul style="list-style-type: none"> <li>• Оценява числената стойност на аритметичен израз, записан на език за програмиране.</li> <li>• Записва аритметичен израз със средствата на език за програмиране.</li> <li>• Прилага и използва вградени в езика за програмиране математически функции – абсолютна стойност, повдигане на степен, закръгляване, извличане на цялата част на реално число.</li> <li>• Използва приоритет на операциите в аритметични изрази, съдържащи вградени функции.</li> <li>• Създава аритметични изрази, съдържащ различни типове данни, като се съобразява със съвместимостта им.</li> <li>• Описва синтаксис и семантика на оператор за присвояване.</li> <li>• Форматира изхода на реално число.</li> </ul>	<ul style="list-style-type: none"> <li>• аритметичен израз в език за програмиране</li> <li>• вградени математически функции в език за програмиране</li> <li>• съвместимост на типове данни</li> <li>• форматиран изход</li> </ul>
<b>3.2 Създаване на компютърна програма за решаване на конкретна задача</b>		

	<ul style="list-style-type: none"> <li>Знае основните етапи при създаване на компютърна програма.</li> <li>Анализира и проектира решението на конкретна задача.</li> <li>Създава математическия модел за решаване на задачата.</li> <li>Разработва алгоритъм за решаване на задачата.</li> <li>Определя входно-изходни данни и техните типове.</li> <li>Структурира и разработва графичен интерфейс, като използва обекти и декларира променливи.</li> <li>Създава и описва програмния код.</li> <li>Стартира, тества и валидира готовия проект.</li> <li>Открива синтактични и логически грешки в програмата.</li> <li>Отстранява синтактични и логически грешки при програмиране.</li> <li>Спазва изисквания за оформяне на програмния код, включващи подравняване, коментари, именуване на програмните единици.</li> </ul>	<ul style="list-style-type: none"> <li>коментари</li> <li>оформяне на програмния код</li> </ul>
<b>3.3. Програмни конструкции за реализация на разклонен алгоритъм</b>		
3.3.1. Булев тип данни	<ul style="list-style-type: none"> <li>Обосновава необходимостта от разклоняване на алгоритмичния процес.</li> <li>Използва константите от булев тип данни.</li> <li>Дава примери, в които се използва булев тип данни.</li> </ul>	<ul style="list-style-type: none"> <li>булев тип данни</li> <li>булеви константи – false, true</li> <li>логически операции</li> <li>приоритет на логическите операции</li> </ul>



	<ul style="list-style-type: none"> <li>• Декларира булева променлива.</li> <li>• Присвоява стойност на булева променлива.</li> <li>• Записва на език за програмиране булев израз, съдържащ операция за сравнение.</li> <li>• Изписва синтактично правилно на езика за програмиране основните логически операции – логическо отрицание, дизюнкция, конюнкция.</li> <li>• Познава приоритета на логическите операции.</li> <li>• Пресмята без използване на компютър стойността на булев израз.</li> <li>• Съставя със средствата на език за програмиране сложен булев израз, отговарящ на дадена логическа ситуация.</li> </ul>	<ul style="list-style-type: none"> <li>• булев израз</li> </ul>
3.3.2. Условен оператор	<ul style="list-style-type: none"> <li>• Описва синтаксиса и семантиката на кратка и пълна форма на условен оператор.</li> <li>• Описва разклонен алгоритъм с помощта на условен оператор.</li> <li>• Използва условен оператор за проверка на коректността на входните данни за програма.</li> <li>• Използва условен оператор за обработка на свойства на радиобутон и поле за отметка.</li> </ul>	<ul style="list-style-type: none"> <li>• условен оператор</li> <li>• съставен оператор</li> </ul>
3.3.3. Вложени условни оператори	<ul style="list-style-type: none"> <li>• Обяснява семантиката на вложени условни оператори в кратка и пълна форма.</li> <li>• Проиграва изпълнението на фрагмент на програма,</li> </ul>	<ul style="list-style-type: none"> <li>• вложен условен оператор</li> </ul>

	<p>съдържаща вложени условни оператори.</p> <ul style="list-style-type: none"> <li>• Записва синтактично и логически правилно вложени условни оператори.</li> <li>• Създава модел и алгоритъм за решаване на задача чрез използване на вложени условни конструкции.</li> <li>• Реализира модел за решаване на задача чрез използване на вложени условни оператори.</li> <li>• Заменя вложен условен оператор с единичен и обратно.</li> </ul>	
<b>3.4. Програмни конструкции за реализация на циклични алгоритми</b>		
3.4.1. Циклични алгоритми	<ul style="list-style-type: none"> <li>• Посочва елементите на циклична конструкция – инициализация, тяло на цикъла и условие на цикъла.</li> <li>• Оценява необходимостта от използването на алгоритми с циклични конструкции с предусловие и постусловие.</li> <li>• Записва синтактично и логически правилно оператори за цикъл с предусловие, постусловие и управлявани от брояч.</li> <li>• Прилага алгоритми с циклични конструкции за проверка на входни данни.</li> <li>• Оценява необходимостта от използване на алгоритми с циклични конструкции с условие или управлявани от брояч.</li> <li>• Преобразува програмен код, съдържащ циклична конструкция, управлявана от брояч в циклична</li> </ul>	<ul style="list-style-type: none"> <li>• структура на циклични алгоритмични конструкции</li> <li>• оператор за цикъл с предусловие</li> <li>• оператор за цикъл с постусловие</li> <li>• оператор за цикъл, управляван от брояч</li> <li>• списъчно поле</li> </ul>

	<p>конструкция или управлявана от условие.</p> <ul style="list-style-type: none"> <li>• Открива синтактични и логически грешки в програмния код на алгоритми с циклична конструкция.</li> <li>• Прилага циклични алгоритми за управление на графичен потребителски интерфейс.</li> </ul>	
3.4.2. Приложение на условни и циклични конструкции	<ul style="list-style-type: none"> <li>• Прилага циклични алгоритми за изчертаване на графични примитиви.</li> <li>• Използва циклични алгоритми за въвеждане и извеждане на данни от файл.</li> <li>• Прилага програмни конструкции за реализация на алгоритми за намиране на сума, минимален/максимален елемент, средно аритметично и др. в редици от числа, въвеждани от потребителския интерфейс/клавиатурата.</li> </ul>	<ul style="list-style-type: none"> <li>• графичен примитиви</li> <li>• кутия за изображения</li> <li>• текстов файл</li> </ul>
<b>3.5. Тестване и верификация на програма</b>		
	<ul style="list-style-type: none"> <li>• Обяснява и разграничава понятията тестване и верификация.</li> <li>• Дефинира тестови данни.</li> <li>• Дефинира очаквани резултати от тестването при определени входни данни.</li> <li>• Използва инструмент за откриване и отстраняване на грешки (debugger).</li> <li>• Прилага процедури за тестване и верификация на вече създадени програми.</li> </ul>	<ul style="list-style-type: none"> <li>• тестване</li> <li>• верификация</li> <li>• тестови данни</li> </ul>
<b>3.6. Съставни типове данни. Едномерен масив</b>		
3.6.1. Едномерен масив	<ul style="list-style-type: none"> <li>• Разбира необходимостта от използване на масиви.</li> </ul>	<ul style="list-style-type: none"> <li>• логическо описание на масив</li> </ul>

	<ul style="list-style-type: none"> <li>• Идентифицира елементите на масив.</li> <li>• Разпознава индекс и стойност на елемент на масив.</li> <li>• Дефинира масив със средствата на език за програмиране.</li> <li>• Създава и инициализира масив със средствата на език за програмиране.</li> <li>• Осъществява достъп до елемент на масив.</li> <li>• Обхожда, въвежда и извежда стойностите на елементите на масив.</li> <li>• Използва списъчно поле за извеждане на стойностите на елементите на масив.</li> </ul>	<ul style="list-style-type: none"> <li>• базов тип на масив</li> <li>• индекс и стойност на елемент от масив</li> </ul>
3.6.2. Основни алгоритми за работа с едномерен масив	<ul style="list-style-type: none"> <li>• Пресмята сбор и произведение на стойностите на елементите на едномерен масив.</li> <li>• Търси елемент от масива с максимална и минимална стойност.</li> <li>• Търси елементи от масива, отговарящи на дадено условие.</li> </ul>	<ul style="list-style-type: none"> <li>• последователно търсене</li> </ul>
<b>4. Създаване на софтуерен проект</b>		
	<ul style="list-style-type: none"> <li>• Описва етапите при реализиране на софтуерен проект.</li> <li>• Извършва проучване и анализ на решения за даден групов проект.</li> <li>• Създава модел за решаване на проблема, поставен в заданието на проекта.</li> <li>• Проектира графичен потребителски интерфейс.</li> </ul>	

	<ul style="list-style-type: none"> <li>• Създава програмен код за реализация на модела.</li> <li>• Създава тестови примери с входни данни и очаквани резултати.</li> <li>• Изготвя документация за софтуерния проект.</li> <li>• Презентира и защитава готовия софтуерен проект.</li> </ul>	
--	---	--

Основната цел на настоящата дипломна работа е да се състави учебно помагало за преподаване на програмиране по представената учебна програма. Помагалото може да бъде ползвано, както от учители, така и от ученици.

### III. Учебно помагало „Въведение в програмирането със C#“

#### 1. Задачите в обучението по програмиране

В тази глава ще срещнете креативни авторски задачи от различно естество, които са пряко свързани с упражнение върху материала, обхващащ учебният план за 8 клас общообразователна подготовка. Задачите са направени така, че да се покаже практическото приложение на изучавания материал, а не само прости примери чрез, които учениците да не могат да разберат как да прилагат изученото. Всички задачи са приложими за различни езици за програмиране, но в тази разработка ще работим изцяло в контекста на езика C# и .NET Core 3.1 платформата, като към момента на разработка това е най-новата и стабилна версия на платформата. За интегрирана среда за разработка на софтуер (IDE – Integrated Development Environment) ще използваме Visual Studio 2019, което е достъпно да бъде свалено от сайта на Microsoft, напълно безплатно (Visual Studio 2019 Community).

#### 2. Бройни системи

Бройната система представлява символен метод за представяне на числата, посредством ограничен брой символи, наречени цифри. Съществуват два вида бройни системи – непозиционни и позиционни.

Непозиционна бройна система наричаме тази бройна система, чиито стойности на всяка цифра не зависят от позицията им. Примери за такива бройни системи са гръцката и римската:

Гръцка цифра	Десетична стойност
Ι	1
Γ	5

Δ	10
Н	100
Х	1000
М	10000

Римска цифра	Десетична стойност
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

Позиционна бройна система наричаме тази бройна система, чиито стойности на всяка цифра зависят от позицията им. Това означава, че стойността на цифрата в числото не е строго определена и зависи от това на коя позиция се намира съответната цифра в дадено число. Примери за такива бройни системи са двоичната, осмичната, десетичната и шестнадесетичната:

Двоична	Осмична	Десетична	Шестнадесетична
0000	0	0	0
0001	1	1	1
0010	2	2	2
0011	3	3	3
0100	4	4	4
0101	5	5	5
0110	6	6	6
0111	7	7	7
1000	10	8	8
1001	11	9	9
1010	12	10	A
1011	13	11	B

1100	14	12	C
1101	15	13	D
1110	16	14	E
1111	17	15	F

Как се образуват числата от десетичната бройна система? За нашият пример ще използваме числото 27865, представяме всяка една цифра, образуваща числото, като започнем от ляво на дясно по следната формула:  $(\text{цифра} * 10^{\text{брой символи} - 1}) + (\text{цифра} * 10^{\text{брой символи} - 2}) + \dots + (\text{цифра} * 10^{\text{брой символи} - n})$

27865 съдържа 5 символа, така че решението на задачата изглежда по следния начин:

$$27865 = (2 \times 10^4) + (4 \times 10^3) + (8 \times 10^2) + (6 \times 10^1) + (5 \times 10^0)$$

За разлика от десетичната бройна система, която е лесно разбираема от човека, двоичната бройна система е по-сложна, но същевременно разбираема за изчислителната машина (компютър). За представяне на число в двоична бройна система се използват само числата 0 и 1. Прието е, когато едно число се записва в бройна система, различна от десетичната, във вид на индекс в долната му част да се отразява, коя бройна система е използвана за представянето му. Например със записа  $1110_{(2)}$  означаваме число в двоична бройна система. Важно е да знаем, че ако едно двоично число завършва на 0, то е четно, а ако завършва на 1, то е нечетно.

Преобразуването на число от двоична в десетична бройна система става, когато всяка една двоична цифра се умножава по 2 на степен, позицията, на която се намира:

$$11001_{(2)} = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 16_{(10)} + 8_{(10)} + 1_{(10)} = 25_{(10)}$$

От това следва, че  $11001_{(2)} = 25_{(10)}$

При преминаване от десетична в двоична бройна система, се извършва преобразуване на десетичното число в двоично. За целите на преобразуването се извършва делене на две с остатък. Така се получават частно и остатък, който се отделя.

$$148 : 2 = 74 \text{ остатък } 0;$$

$$74 : 2 = 37 \text{ остатък } 0;$$

$$37 : 2 = 18 \text{ остатък } 1;$$

$$18 : 2 = 9 \text{ остатък } 0;$$

$$9 : 2 = 4 \text{ остатък } 1;$$

$$4 : 2 = 2 \text{ остатък } 0;$$

$$2 : 2 = 1 \text{ остатък } 0;$$

$$1 : 2 = 0 \text{ остатък } 1;$$

След като вече сме извършили деленето, записваме стойностите на остатъците в ред, обратен на тяхното получаване, както следва:

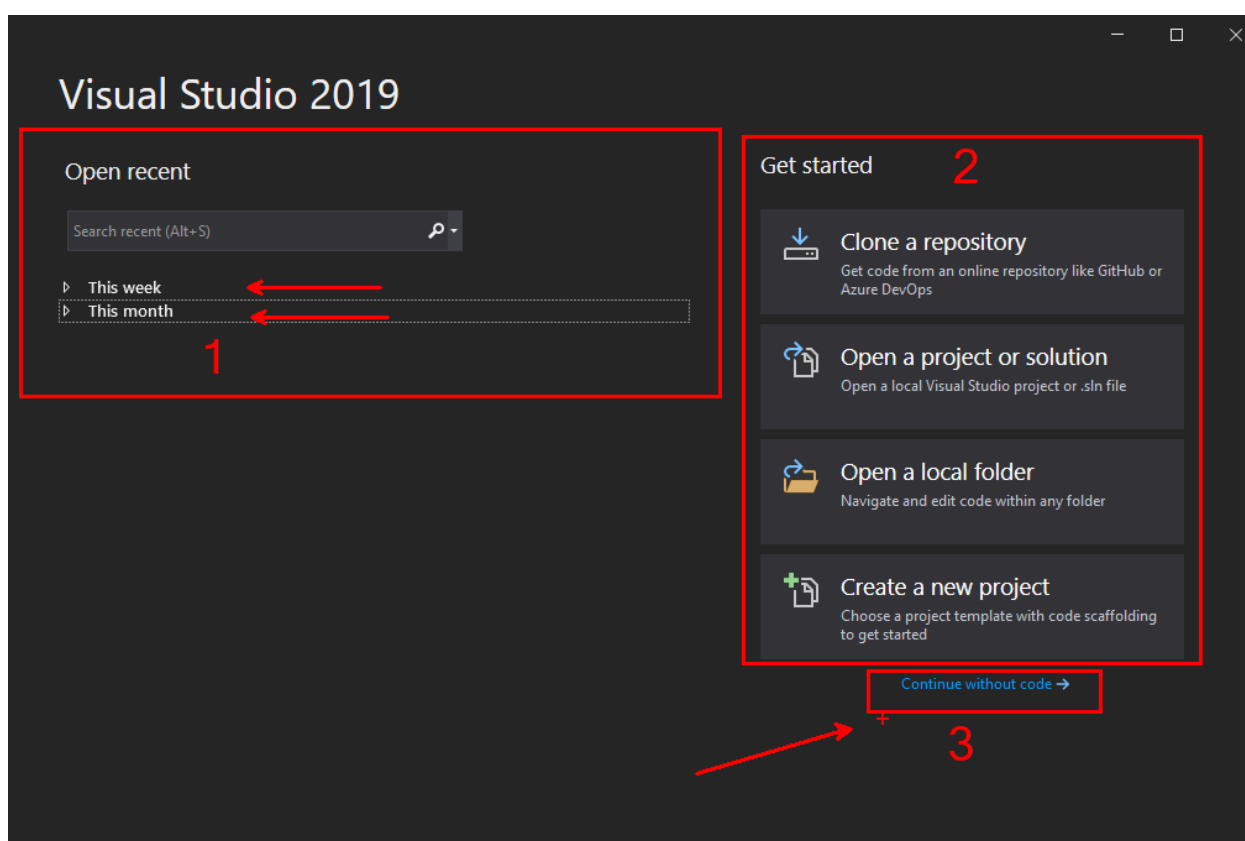
$$148_{(10)} = 10010100_{(2)}$$

Аритметични правила за събиране, изваждане и умножение на двоични числа:

$0 + 0 = 0$	$0 - 0 = 0$	$0 \times 0 = 0$
$1 + 0 = 1$	$1 - 0 = 1$	$1 \times 0 = 0$
$0 + 1 = 1$	$1 - 1 = 0$	$0 \times 1 = 0$
$1 + 1 = 10$	$10 - 1 = 1$	$1 \times 1 = 1$

### 3. Интегрирана среда за разработка (IDE)

Интегрираната среда за разработка представлява софтуерно приложение, което разполага с пълния набор от инструменти нужни на програмиста да създаде собствено софтуерно приложение. То се състои от няколко основни компонента: редактор на код, компилатор/интерпретатор, дебъгер, система за контрол на версиите. В контекста на тази разработка интегрираната среда за разработка, която ще използваме е Visual Studio 2019. След като сте изтеглили и инсталирали Visual Studio 2019, може да преминем към запознаване с неговият интерфейс.



След отваряне на Visual Studio 2019, попадаме на началния екран, където има три основни секции. В секцията, номерирана с номер 1 на снимката, виждаме последно отваряните проекти, в секция номер 2 са някои основни преки пътища, които за момента няма да използваме. Секция номер 3 е желаната от нас - натискаме върху

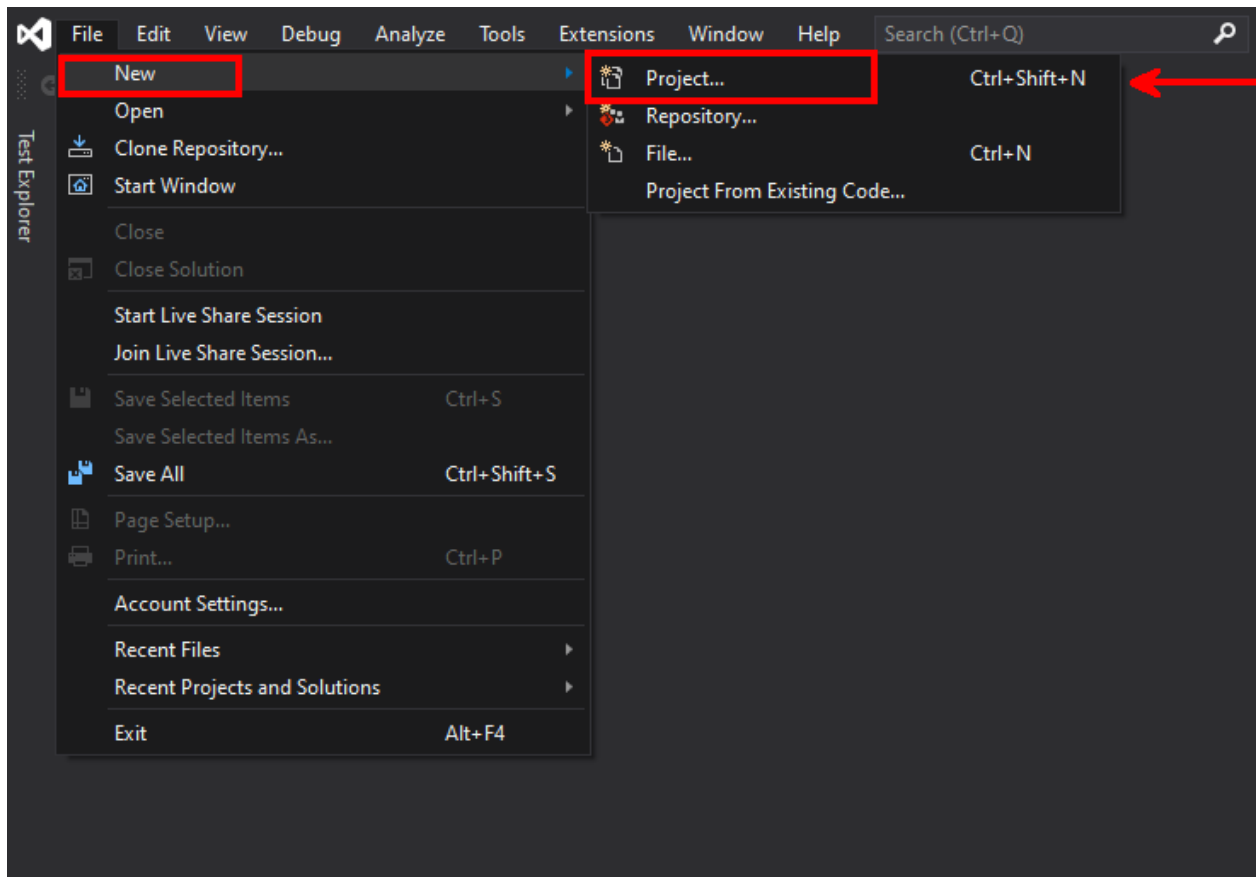


надписа. Попадаме в един празен екран, забелязваме, че интегрираната среда за разработка наподобява много на обикновена програма тоест има менюта и бутони. Основни секции на интегрираната среда за разработка:

1. стандартно навигационно меню (navigation menu)
2. често използвани функции на средата и преки пътища (shortcuts)
3. файловете на решението (solution explorer)
4. списък с компилационни грешки и предупреждения (errors and warnings)
5. текстови редактор на кода (text editor)

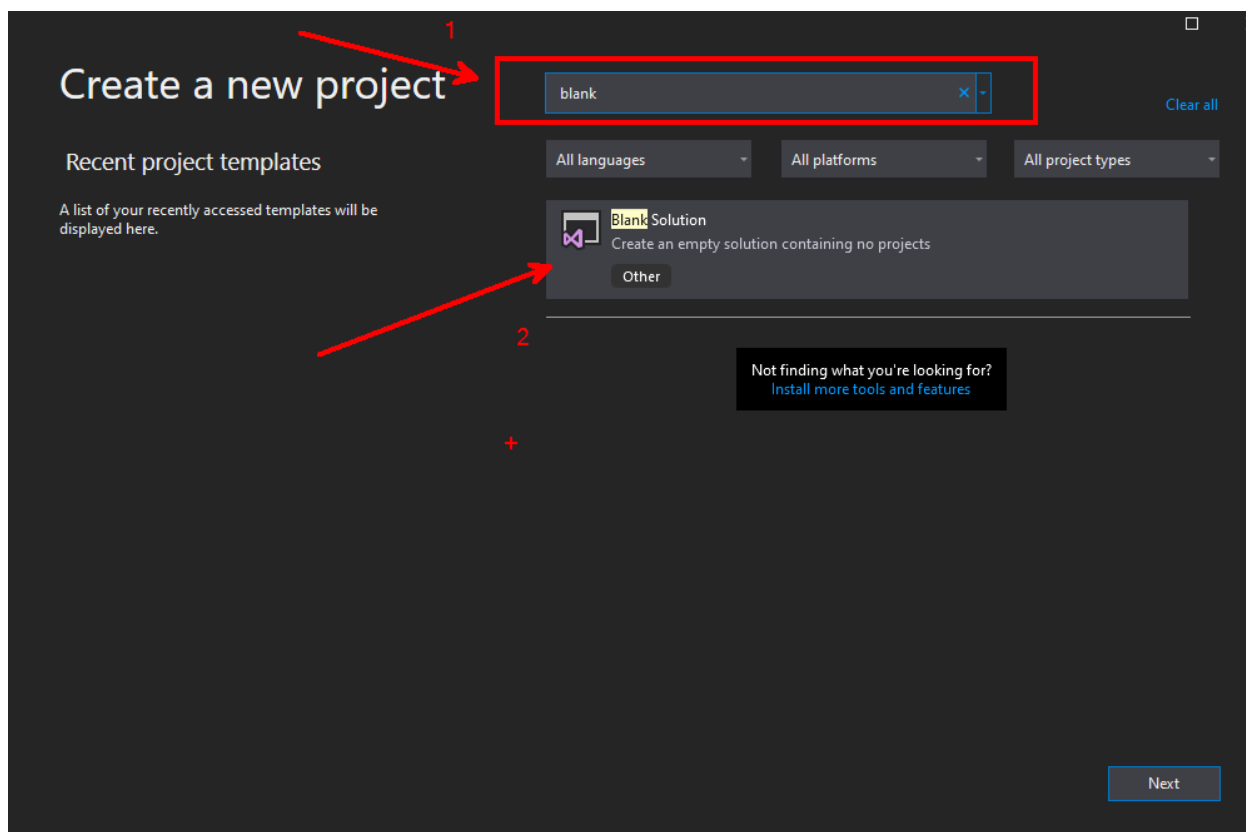


## 4. Създаване на решение (Solution)

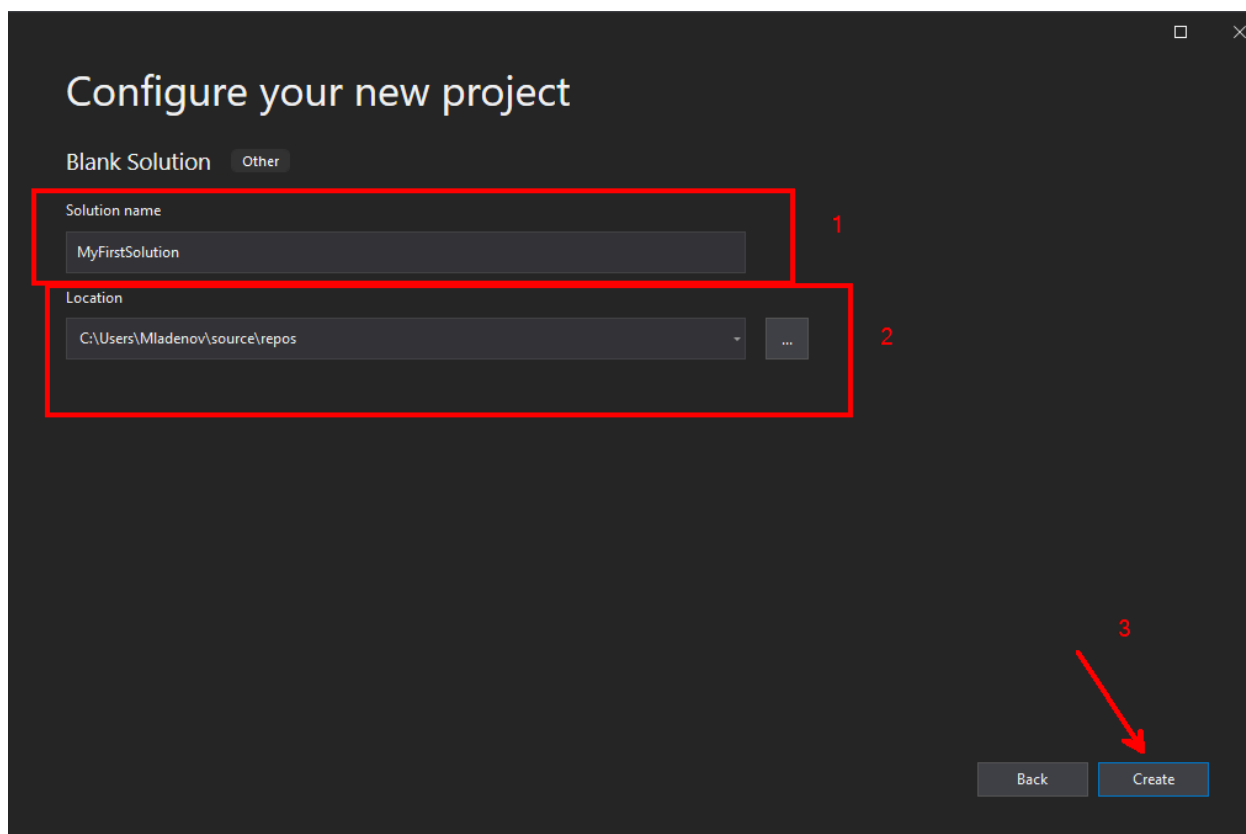


След като сме в интегрираната среда за разработка, от навигацията горе в ляво избираме File → New → Project.

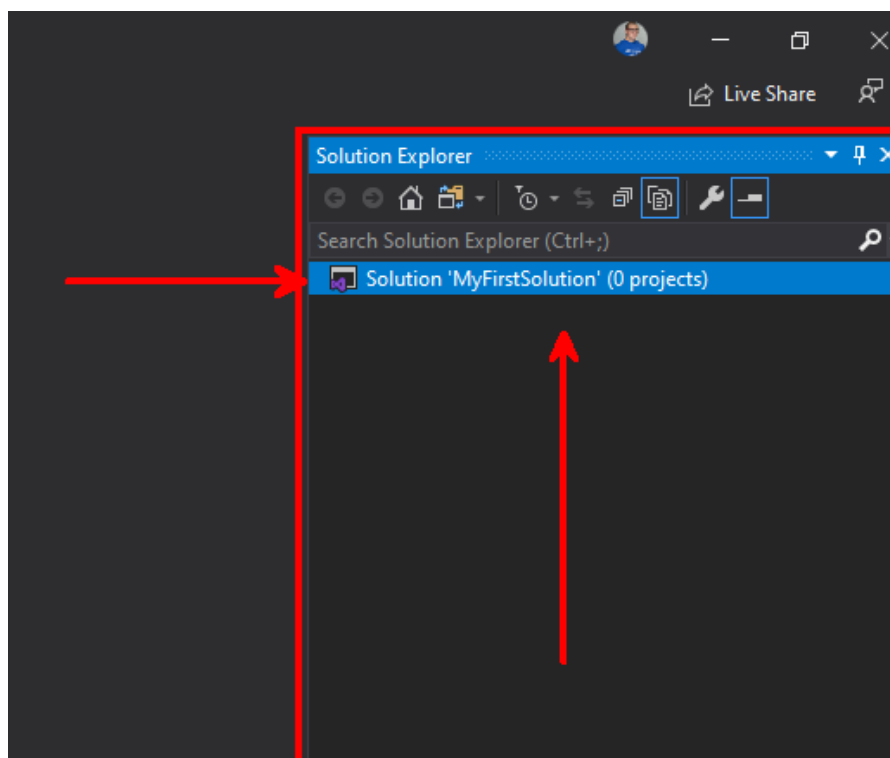
В отворения прозорец има търсачка - в нея изписваме „Blank“ и средата ни показва предложение за „Blank Solution“, селектираме го и натискаме бутона „Next“.



В първата секция слагаме говорещо за решението име, за целта на демото го именуваме „MyFirstSolution“. От втората секция избираме местоположението на решението. След като сме готови натискаме бутона „Create“



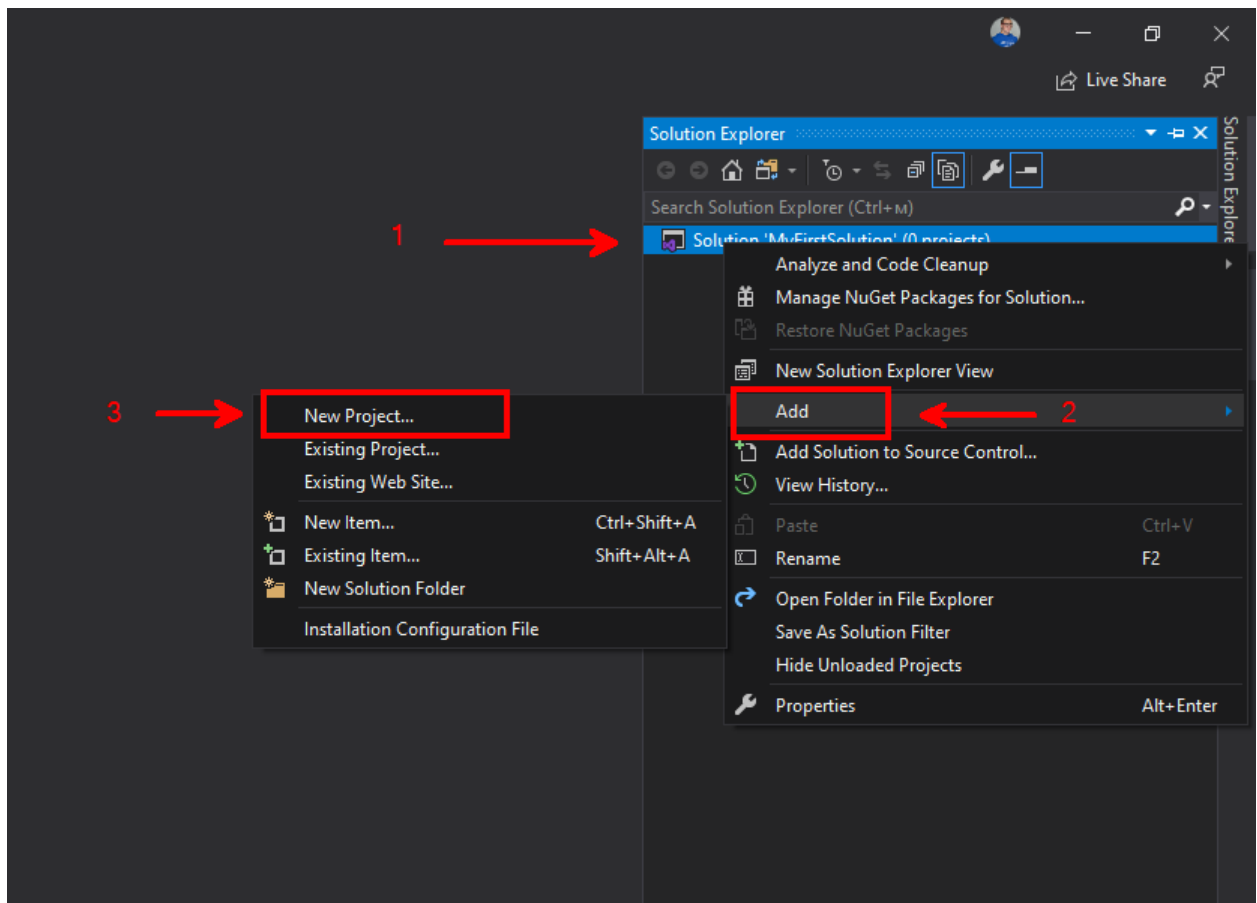
В „Solution Explorer“ може да видим, че нашето празно решение е създадено успешно.



## 5. Първа програма

В това упражнение ще се запознаем с това, как да създадем нашата първа програма, чиято цел ще бъде изписването на текст в конзолата. Ще разберем още как се стартира тя и някои важни концепции относно именуването и.

След като вече имаме създадено решение, а именно решението от предишното упражнение, което беше именувано „MyFirstSolution“, вече може да добавим първата ни програма към него.



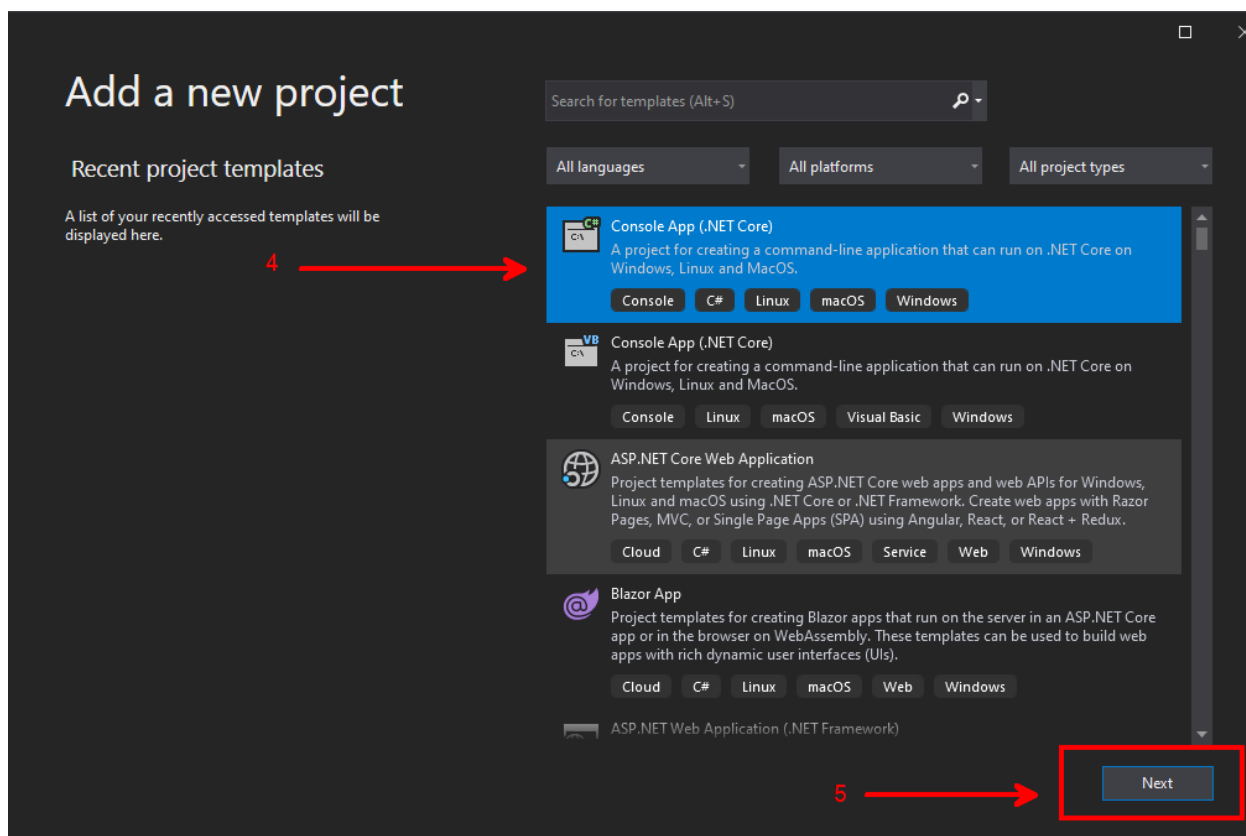
1 – натискаме десен бутон на мишката върху решението

2 – избираме от менюто Add

3 – натискаме върху New Project

4 – избираме Console App (.NET Core)

5 – натискаме Next



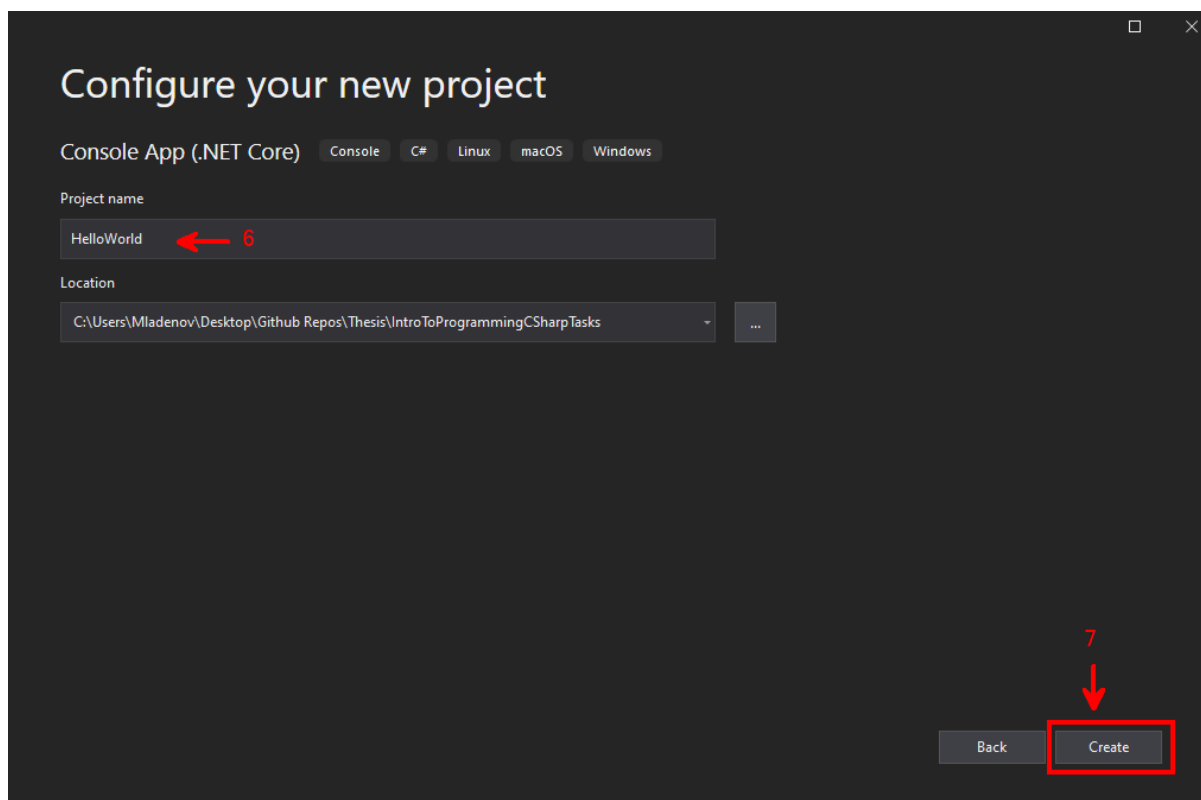
6 – на полето „Project name“ трябва да обърнем малко повече внимание. Име специална конвенция за именуване на нашите проекти, която гласи следното: проект или решение винаги се именуват с описателни имена, като първата буква винаги е главна, ако името съдържа няколко думи никога не се слагат паузи, вместо паузи изписваме всяка следваща дума, долепена до предходната, като първата ѝ буква трябва да е главна.

Примери:

- грешен пример – „Hello world program exercise“
- верен пример – „HelloWorldProgramExercise“

7 – Натискаме бутона Create

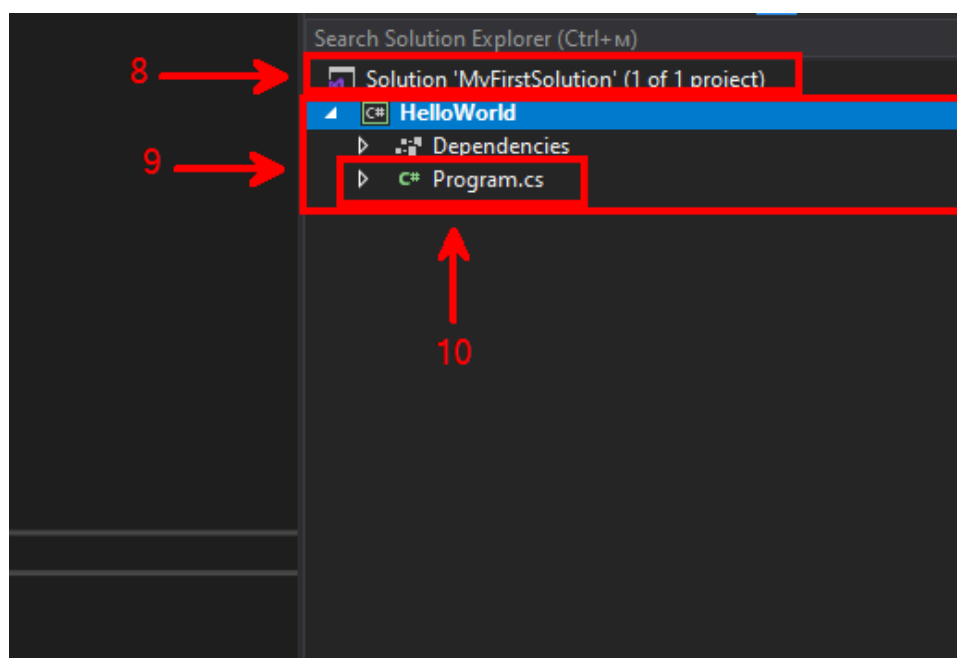
Важно е да споменем, че когато добавяме програма във вече създадено решение, както е в случая, не променяме местоположението му (Location), за да бъде създадена програмата вътре във вече създаденото решение!

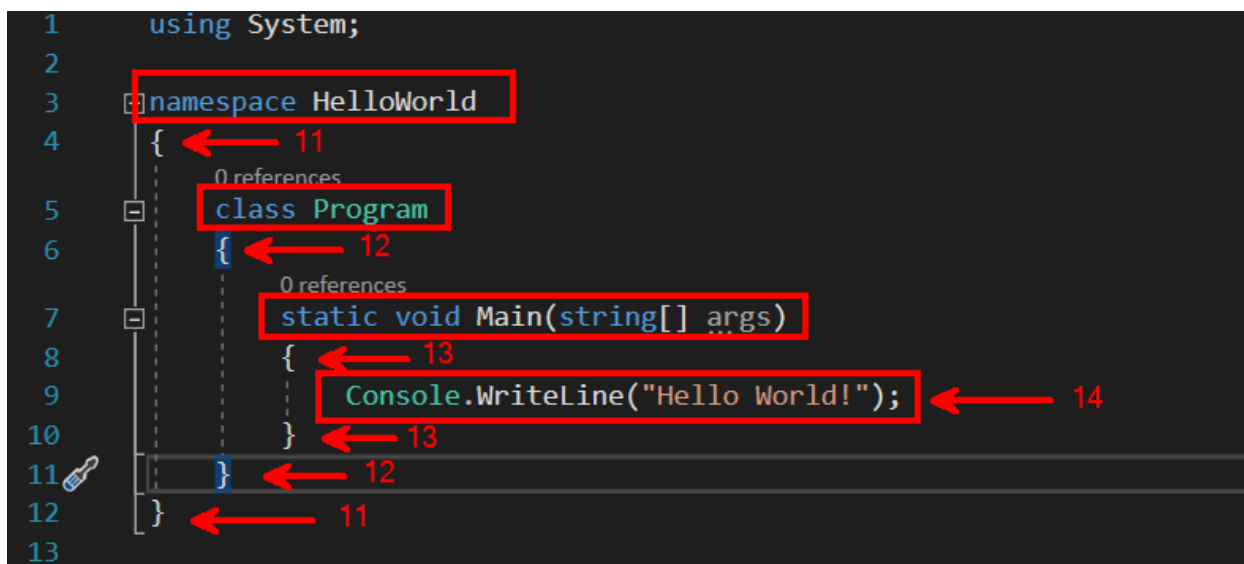


8 – както вече знаем това е решението

9 – програмата, която създадохме

10 – файла, в който ще въвеждаме кода на програмата ни





11 – отваряща и затваряща скоба на namespace HelloWorld, което означава, че това е обхвата (scope) на този namespace или казано по-просто, обхвата на програмата, която създадохме с име „HelloWorld“, който отговаря на секцията от 9-та стъпка

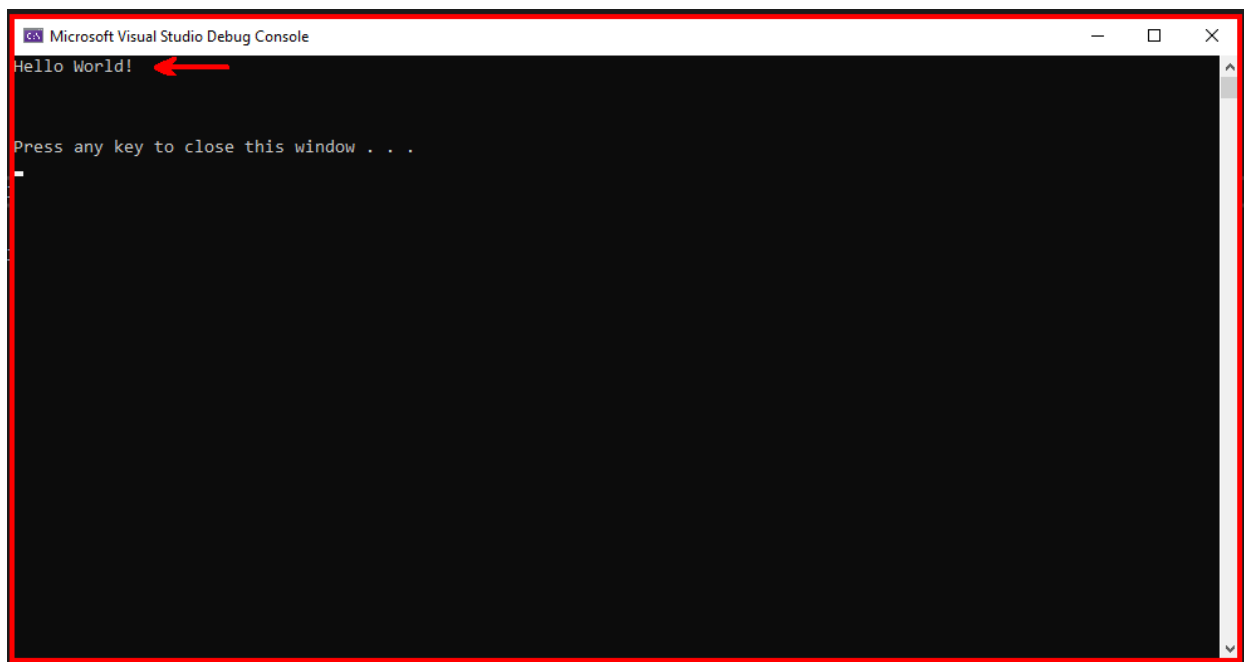
12 – отваряща и затваряща скоба на class Program, също както на предходната стъпка, двете скоби дефинират обхвата на класа Program, който съответства на секцията от стъпка 10

13 – отваряща и затваряща скоба на метода Main, както при предходните два примера скобите и тук дефинират обхвата на метода Main, за методи ще говорим по-късно в тази разработка, за момента е нужно да знаем, че всички код, който искаме да бъде изпълнен, трябва да се намира между отварящата и затварящата скоба на Main метода, защото той е началната точка при стартиране на конзолно приложение

14 – секция за писане на логически код

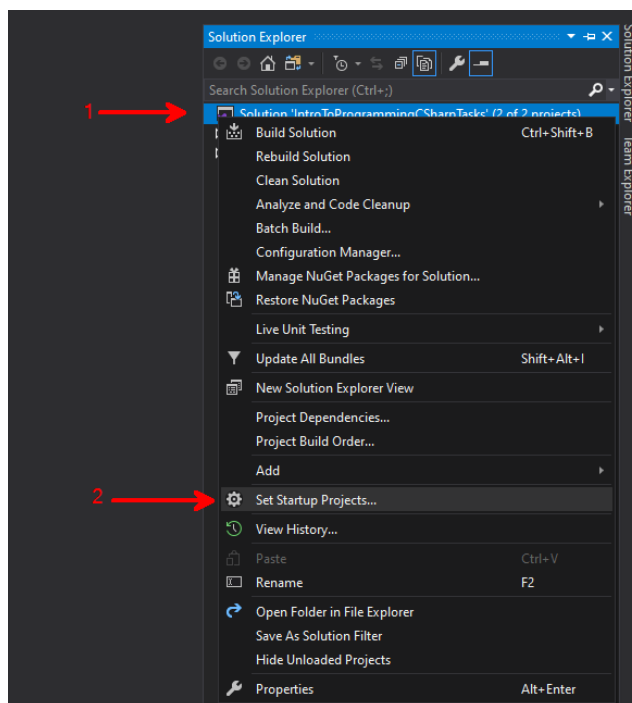
Нека се фокусираме върху последната секция и разгледаме по-подробно кода, който се състои в случая от един ред, но същевременно ще изпълни заданието ни. Ако разбираме английски език, командите които са изписани, ще ни се сторят доста описателни и естествени. Буквално казваме на компютъра: „Конзола, напиши ред, който да съдържа следният текст → „Hello World!“. Дотук сме се справили със задачата, но никъде не виждаме нито конзола, нито текст, това е защото нашата програма все още не е стартирана, за да я стартираме трябва да натиснем клавишната комбинация Ctrl + F5. След натискане на тази клавишна комбинация, програмата се компилира, тоест проверява се за грешки, предупреждения или други проблеми, които могат да възпрепятстват нейното изпълнение и се стартира след това. Желаният резултат от изпълнението ѝ трябва да изглежда по следния начин:

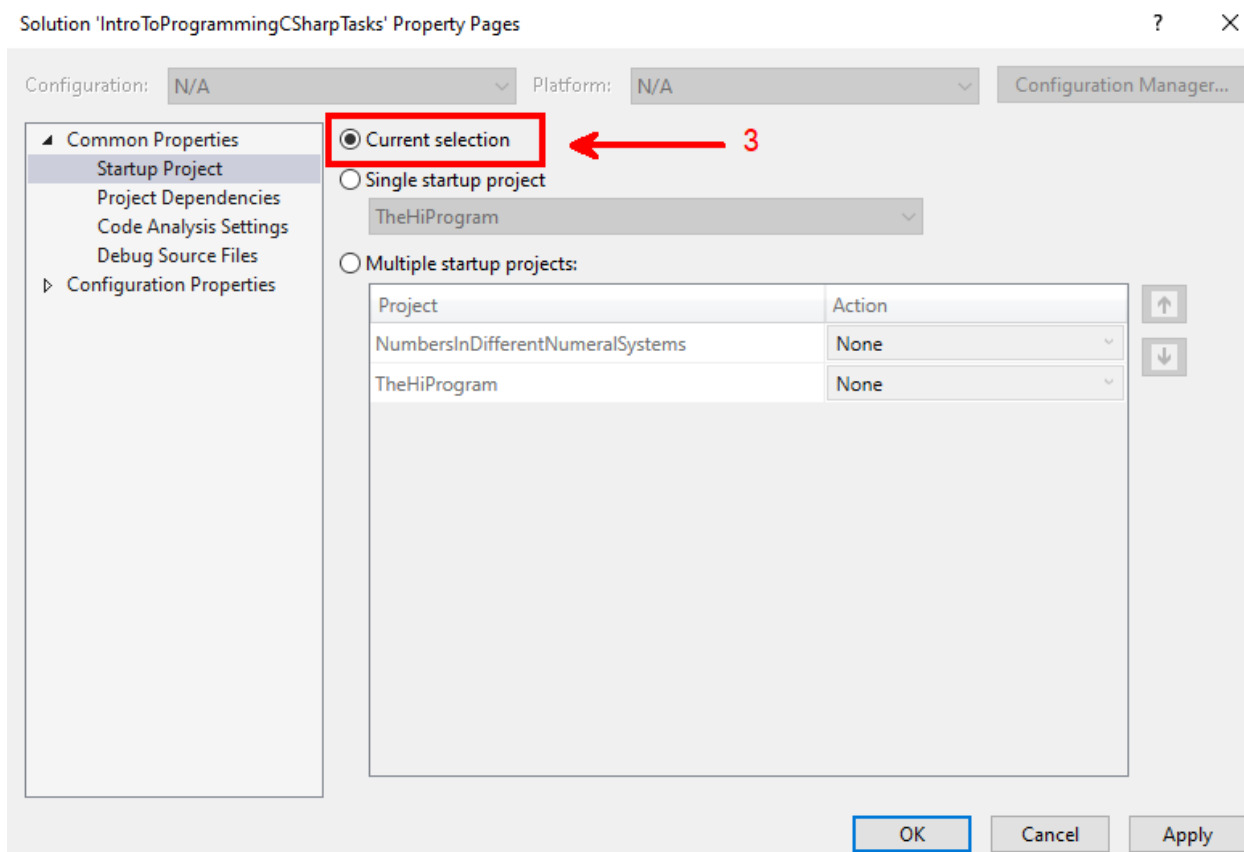




Може да се поздравим, понеже програмата изпълни исканията ни напълно, нямаше грешки или предупреждения. Ако за момента това Ви се струва малко, ако изпълните всички задания до края на курса, ще се научите да правите далеч по-сложни и полезни приложения. Нека обобщим наученото с решението на няколко задачи.

Когато имаме повече от един проект в нашето решение и искаме при натискане на комбинацията Ctrl + F5 да се стартира проекта, по когото работим в момента, трябва да направим следната настройка:





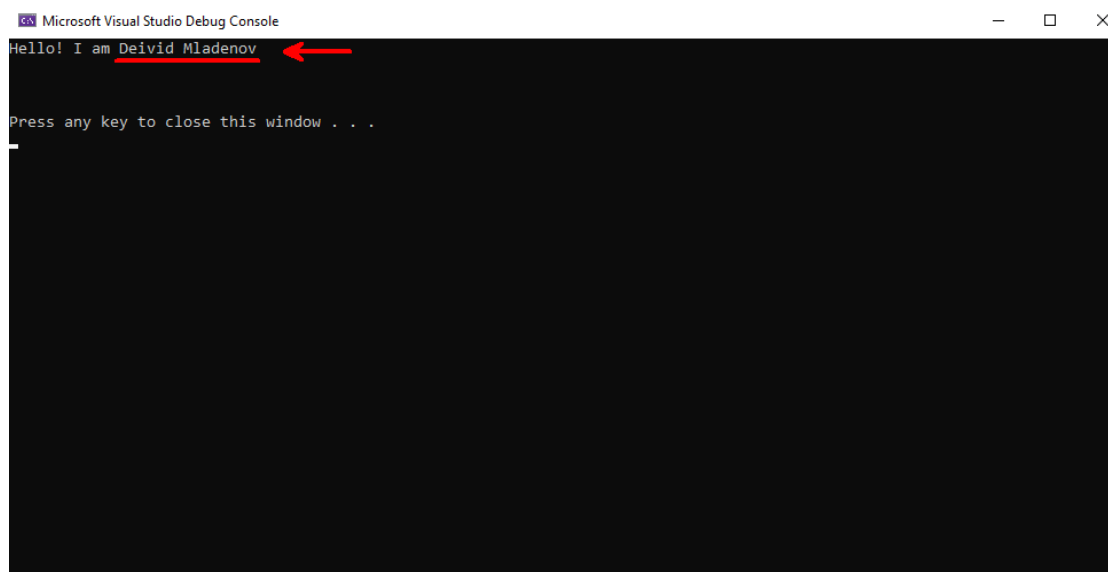
- 1 – десен бутон върху решението
- 2 – избираме Set Startup Project
- 3 – отбелязваме Current selection и натискаме OK

**Задача 1:** Създайте решение (Solution), в което ще събирате всички задачи от обучението в тази разработка. Именувайте го по следния начин: „IntroToProgrammingCSharpTasks“.

### Задача 2: TheHiProgram

Създайте конзолно приложение, което изписва следния текст: „Hello! I am <вашето име>“.

Желано поведение на програмата:



## 6. Примитивни типове данни и променливи

Всички модерни програми в днешно време съдържат променливи в кода си. Най-просто казано, променливата (variable) представлява резервирано място в оперативната памет на компютъра, където ние можем да запазваме данни от различни видове (числа, символи, текст и т.н). От името им може да разберем, че тази стойност може да бъде променяна по време на изпълнение на програмата.

Типовете данни представляват диапазони от стойности, които имат еднакви характеристики. Съществуват няколко групи примитивни типове данни в C#, те се наричат примитивни защото са част от езика:

- Целочислени – sbyte, byte, short, ushort, uint, int, long, ulong
- Реални с плаваща запетая – float, double
- Реални с десетична точност – decimal
- Булев – bool
- Символен – char
- Символен низ – string
- Обект – object

Тип данни	Стойност по подразбиране	Минимална стойност	Максимална стойност
sbyte	0	-128	127
byte	0	0	255
short	0	-32768	32767
ushort	0	0	65535
int	0	-2147483648	2147483647
uint	0u	0	4294967295
long	0L	-9223372036854775808	9223372036854775807
ulong	0u	0	18446744073709551615
float	0.0f	$\pm 1.5 \times 10^{-45}$	$\pm 3.4 \times 10^{38}$
double	0.0d	$\pm 5.0 \times 10^{-324}$	$\pm 1.7 \times 10^{308}$
decimal	0.0m	$\pm 1.0 \times 10^{-28}$	$\pm 7.9 \times 10^{28}$
bool	false	Има само две стойности true и false	
char	'\u0000'	'\u0000'	'\uffff'
string	null	-	-
object	null	-	-

Декларирането на променлива става по указаният начин, за целта на примера ще деклариране няколко променливи от различен тип:

```

0 references
static void Main(string[] args)
{
    1  int number = 12;
    2
    3
    4
    5
    6  double floatingPointNumber = 2.25555;
    7
    8  string helloWorldText = "Hello world!";
    char comma = ',';
}

```

The image shows a code editor with C# code. Red arrows and numbers 1 through 8 point to specific parts of the code, illustrating the steps for declaring variables. Arrows 1-5 point to the declaration of 'int number = 12;'. Arrows 6-8 point to the declarations of 'double floatingPointNumber = 2.25555;', 'string helloWorldText = "Hello world!";', and 'char comma = \',\';' respectively.

В числата от едно до пет на схемата, са показани стъпките за деклариране на променлива:

- 1 – тип данни
- 2 – говорещо за променливата име
- 3 – знак равно, който означава присвояване на стойност
- 4 – началната стойност, която искаме да притежава нашата променлива
- 5 – знак точка и запетая, който означава край на декларацията

В останалите числа на схемата ще се запознаем с някои особености и конвенции:

6 – променливите в C# винаги се именуват с малка буква, ако името, за да бъде достатъчно описателно, е нужно да съдържа повече от една дума, то тогава всяка следваща дума се изписва с главна буква. В имена на променливи не се допуска слагане на паузи, служебни думи от езика или числа!

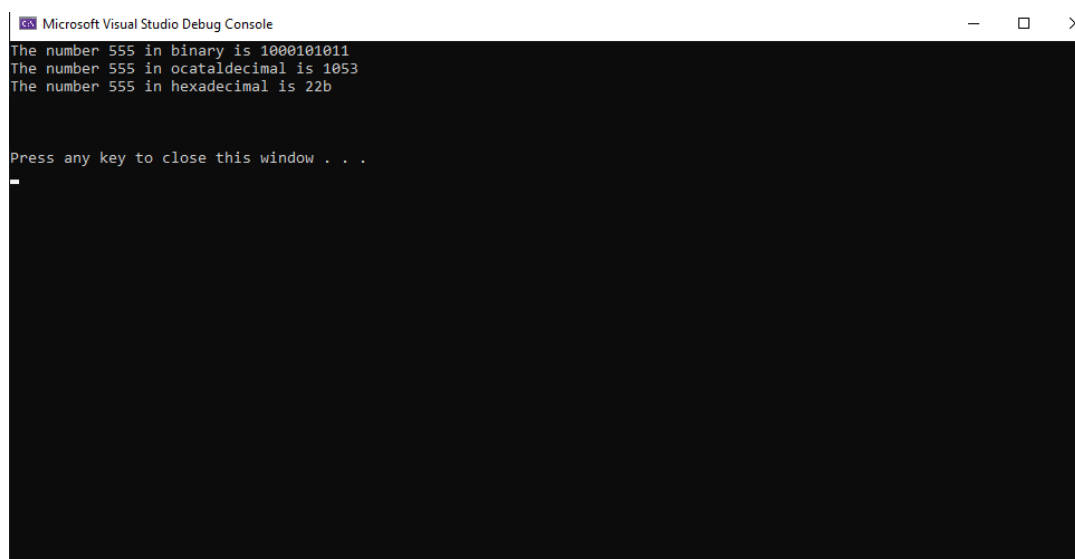
7 – двойни кавички използваме при декларация на символен низ

8 – единични кавички използване при декларация на един символ

### Задача 3: NumbersInDifferentNumeralSystems

Създайте конзолно приложение, което пресмята стойността на числото 555 в двоична, осмична и шестнадесетична бройна система и изписва резултатите в конзолата.

Желано поведение на програмата:



```
Microsoft Visual Studio Debug Console
The number 555 in binary is 1000101011
The number 555 in octal is 1053
The number 555 in hexadecimal is 22b

Press any key to close this window . . .
```

## 7. Оператори

Операторите позволяват обработката на примитивни типове данни и обекти. Те представляват специални символи и се делят на няколко основни вида:

- аритметични
- логически

- побитови
- за сравнение
- за присвояване
- за преобразуване на типовете

Категория	Оператори
аритметични	-, +, *, /, %, ++, --
логически	&&,   , !, ^
побитови	&,  , ^, ~, <<, >>
за сравнение	==, !=, >, <, >=, <=
за присвояване	=, +=, -=, *=, /=, %=, &=,  =, ^=, <<=, >>=
съединяване на символни низове	+
за работа с типове	(type), as, is, typeof, sizeof
други	., new, (), [], ?:, ??

Също както в математиката и в програмирането има приоритет на операциите, в таблицата долу са показани операциите с най-висок приоритет към операциите с най-нисък.

Приоритет	Оператори
най-висок	++, --, new, (type), typeof, sizeof
	++, --, +, -, !, ~
	*, /, %
	+ (за свързване на низове)
	+, -
	<<, >>
	<, >, <=, >=, is, as
	==, !=
	&, ^,
	&&

най-нисък	?:, ??
	=, *=, /=, %=, +=, -=, <<=, >>=, &=, ^=,  =

Логическите оператори работят с булеви стойности (true и false).

Оператор	Значение
&&	логическо и
	логическо или
!	логическо отрицание
^	изключващо или

## 8. Вход и изход от конзолата

До момента ни се е налагало да разпечатваме резултатите от задачите в конзолата. В тази тема ще се упражним, освен да разпечатваме нещо на нея, а и как да четем информация от нея. По този начин нашите програми, ще могат да водят така наречения „диалог“ с потребителя.

Команда	Действие
Console.Write();	Изписва подадения текст в конзолата, но не преминава на нов ред
Console.WriteLine();	Изписва подадения текст в конзолата, като преминава на нов ред
Console.Read();	Прочита следващия натиснат символ
Console.ReadLine();	Прочита целия ред от символи
Console.ReadKey();	Прочита кой бутон е бил натиснат

### Задача 4: AgeAfter10Years

Създайте конзолно приложение, което пресмята възрастта Ви след десет години и изписва резултатите в конзолата. Вашата програма трябва да извършва „диалог“ с потребителя, като го пита за неговата възраст и той трябва да я въведе. Програмата трябва да извършва правилни сметки независимо от подаденото число!

Вход	Изход
22	You will be 32 years old after 10 years!
50	You will be 60 years old after 10 years!

Желано поведение на програмата:

```

Microsoft Visual Studio Debug Console
How old are you?
22
You will be 32 years old after 10 years!

Press any key to close this window . . .

```

### Задача 5: TheUniqueNumber

Създайте конзолно приложение, което пресмята вашето уникално число. Формулата за пресмятане на вашето уникално число е:

**уникално число = (възрастта Ви след 10 години \* годината Ви на раждане) /  $\pi$  \* номера на месеца Ви на раждане**

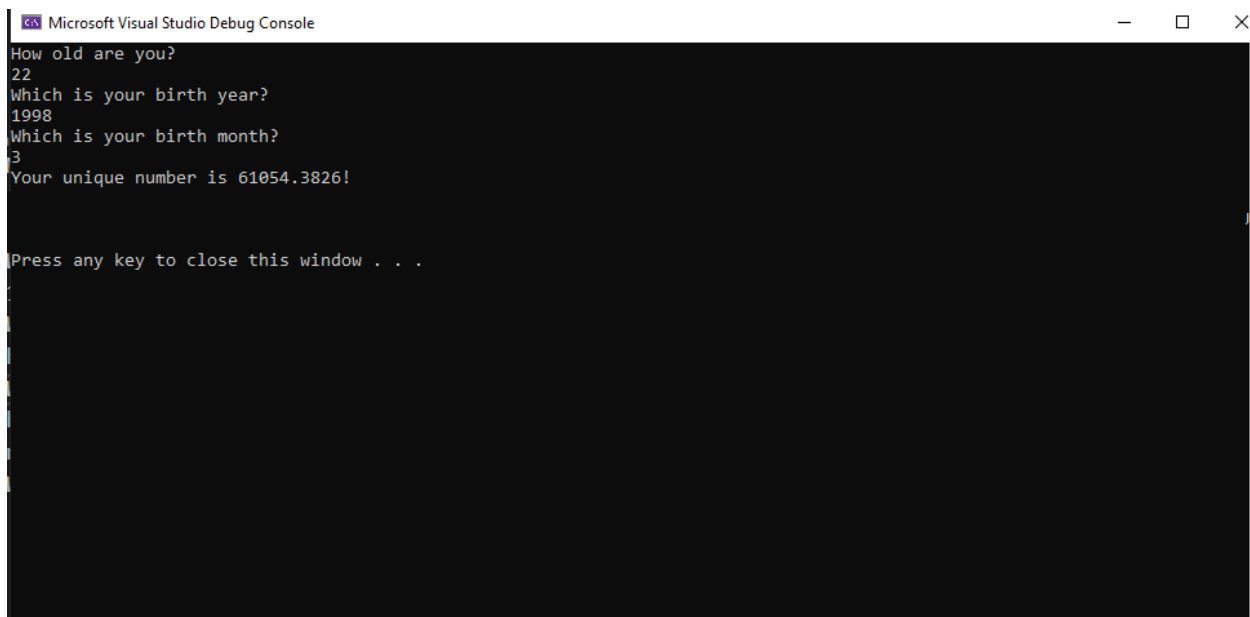
Резултата трябва да бъде закръглен до 4 знак след десетичната запетая. Резултата трябва да бъде разпечатан в конзолата.

Подсказка: Проучете в интернет относно приложението на класа Math в C#.

Вход	Изход
22	Your unique number is 61054.3826!
1998	
3	



Желано поведение на програмата:



```
Microsoft Visual Studio Debug Console
How old are you?
22
Which is your birth year?
1998
Which is your birth month?
3
Your unique number is 61054.3826!

Press any key to close this window . . .
```

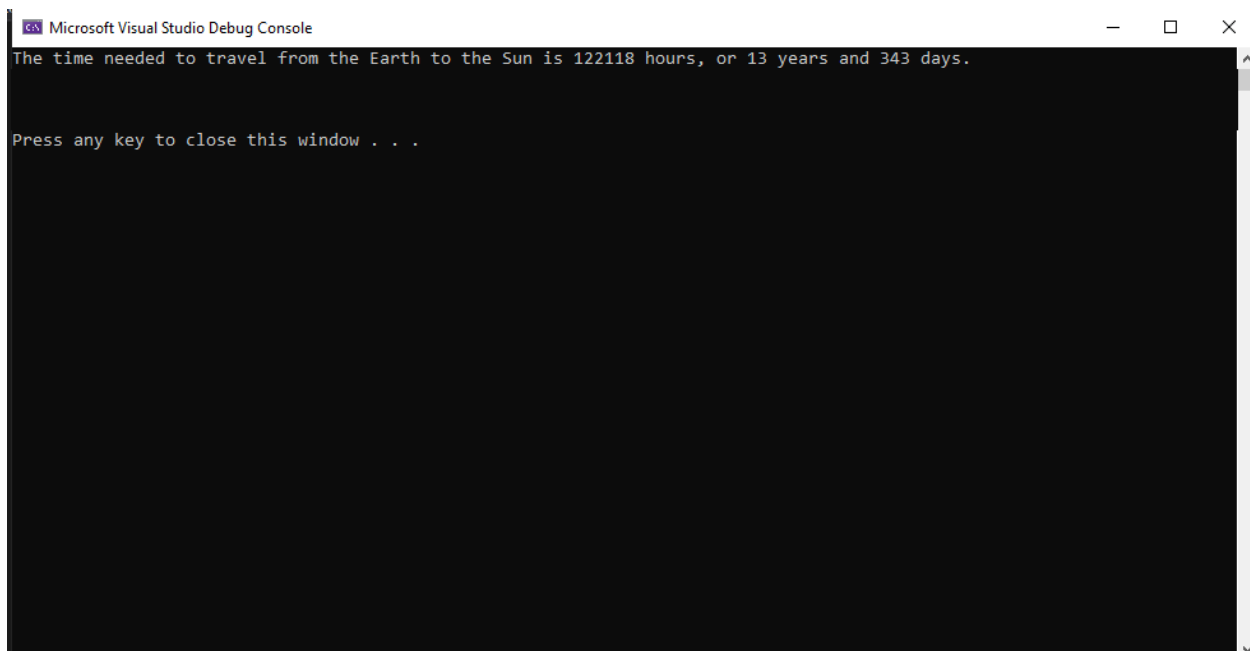
### Задача 6: TravelingToTheSunWithSoundSpeed

Създайте конзолно приложение, което пресмята нужното време, за да стигнете до слънцето, ако пътувате със скоростта на звука. Вашата програма трябва да отпечатва резултата в конзолата.

Разстояние до слънцето: 152080000 км

Скоростта на светлината: 1234.8 км/ч

Желано поведение на програмата:



```
Microsoft Visual Studio Debug Console
The time needed to travel from the Earth to the Sun is 122118 hours, or 13 years and 343 days.

Press any key to close this window . . .
```

## 9. Условни конструкции

Съществуват три основни вида условни конструкции в C#, а именно if, if-else и switch case. Условната конструкция от типа if изглежда по следния начин:

```
if (булев израз)
{
    тяло на условната конструкция;
}
```

Оформлението ѝ включва if-клауза, булев израз и тяло на условната конструкция.

Условната конструкция от типа if-else изглежда по следния начин:

```
int x = 2;
if (x > 3)
{
    Console.WriteLine("x е по-голямо от 3");
}
else
{
    Console.WriteLine("x не е по-голямо от 3");
}
```

Оформлението ѝ също включва if-клауза, булев израз и тяло на условната конструкция, но ако условието в if-клаузата не бъде изпълнено, ще се изпълни else-клаузата.

Важно е да се знае, че if и if-else условни конструкции могат да се влагат едни в други, или казано с прости думи, в тялото на една if-условна конструкция, може да имаме if-else-условна конструкция например.

Условната конструкция от типа switch case изглежда по следния начин:

```
switch (селектор)
{
    case целочислена-стойност-1: конструкция; break;
    case целочислена-стойност-2: конструкция; break;
    case целочислена-стойност-3: конструкция; break;
    default: конструкция; break;
}
```

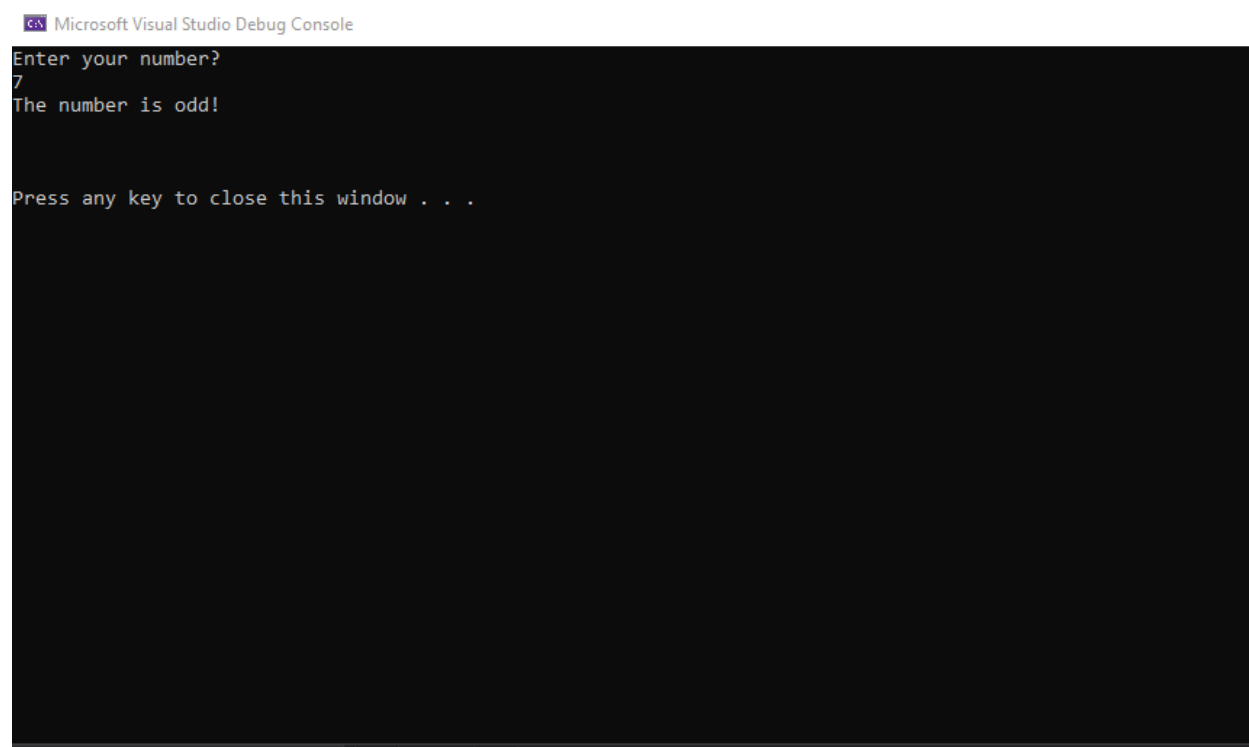
Оформлението ѝ включва switch-оператора, селектор, който е израз, връщащ като резултат някаква стойност, която може да бъде сравнявана, например число или string. Ако се открие съвпадение с някой case етикет, се изпълнява съответната конструкция. Ако не се открие съвпадение, се изпълнява default конструкцията.

### Задача 7: OddOrEven

Създайте конзолно приложение, което при зададен вход в конзолата, който е число, програмата отпечатва дали въведеното число е четно или нечетно. Препоръчително е вашата програма да извършва адекватен „диалог“ с потребителя, както е показано в примера долу. Вашата програма трябва да работи за всички подадени числа, независимо от големината им!

Вход	Изход
7	The number is odd!
2	The number is even!

Желано поведение на програмата:



```

Microsoft Visual Studio Debug Console
Enter your number?
7
The number is odd!

Press any key to close this window . . .

```

### Задача 8: NumberComparison

Създайте конзолно приложение, което получава като вход от потребителя две числа. След това изписва в конзолата дали първото число е по-голямо, по-малко или равно на второто. Вашата програма трябва да работи за всички подадени числа, независимо от големината им!

Вход	Изход
222 45	222 is greater than 45!
55 55	55 is equal to 55!
45 222	45 is less than 222!

Желано поведение на програмата:

```

Microsoft Visual Studio Debug Console
Enter first number:
55
Enter second number:
55
55 is equal to 55!

Press any key to close this window . . .

```

### Задача 9: MyNumberDiapason

Създайте конзолно приложение, което получава като вход от потребителя число. След това отпечатва в конзолата в кой от трите диапазона попада. Трите диапазона са следните:

1. въведеното число е по-малко от 1000
2. въведеното число е между 1000 и 10000
3. въведеното число е по-голямо от 10000

Вход	Изход
888	The number 888 is less than 1000!
5000	The number 5000 is between 1000 and 10000!
15000	The number 88 is greater than 10000!

Желано поведение на програмата:

```

Microsoft Visual Studio Debug Console
Enter your number:
5000
The number 5000 is between 1000 and 10000!
Press any key to close this window . . .

```

### Задача 10: CalculateMyWeightInSolarSystem

Създайте конзолно приложение, което получава като вход от потребителя число, което отговаря на номера на предефиниран списък с планетите от слънчевата система. След въведен номер на планетата, вашата програма трябва да пита за теглото на потребителя в килограми. Когато потребителя въведе цялата информация, вашата програма трябва да пресметне теглото на човека за избраната от него планета от слънчевата система.

Важно е да имате в предвид, че теглото се определя по формулата:

теглото в килограми в на избраната планета = гравитацията на съответната планета \* вашето тегло

Гравитацията на планетите в таблицата са закръглени до втория знак след десетичната запетая!

Планета	Гравитация
Меркурий	0,38
Венера	0,91
Земя	1
Марс	0,38
Юпитер	2,36
Сатурн	0,92
Уран	0,89
Нептун	1,12
Плутон	0,07

Желано поведение на програмата:

```

Microsoft Visual Studio Debug Console
-----
Enter the code corresponding to the name of the desired planet:
1: Mercury
2: Venus
3: Earth (You already know the answer)
4: Mars
5: Jupiter
6: Saturn
7: Uranus
8: Neptune
9: Pluto
-----
1
Enter your weight in kgs as measured on Earth:
100
You weigh approximately 38kg on Mercury.

Press any key to close this window . . .

```

## 10. Цикли

В програмирането често се налага многократно изпълнение на дадена последователност от операции. Цикъл (loop) е основна конструкция в програмирането, която позволява многократно изпълнение на даден фрагмент сорс код. В зависимост от вида на цикъла, програмният код в него се повтаря или фиксиран брой пъти, или докато е в сила дадено условие. Цикъл, който никога не завършва, се нарича безкраен цикъл (infinite loop). Използването на безкраен цикъл рядко се налага освен в случаи, когато някъде в тялото на цикъла се използва операторът break, за да бъде прекратено неговото изпълнение преждевременно.

Съществуват два основни вида цикли в C#, цикли с предусловие (условието за изпълнение на цикъла се проверява дали е валидно преди да се изпълни) и цикли със следусловие (условието за изпълнение на цикъла се проверява дали е валидно след първото завъртане).

Конструкция и начин на работа на циклите в C#:

1. **while** – цикъла ще повтаря докато логическия израз в кръглите скоби не стане със стойност false

```
while (условие)
{
    желан сорс код за изпълнение;
}
```

2. **do while** – цикъла ще се изпълни един път и след това ще провери стойността на логическия израз в кръглите скоби, ако е true ще продължи да се върти, докато не стане false, а ако е false ще прекрати изпълнението си

```
do
{
    желан сорс код за изпълнение;
}
while (израз);
```

3. **for** - се състои от инициализационна част за брояча (int i = 0), булево условие (i < 10), израз за обновяване на брояча (i++, може да бъде i-- или например i = i + 3) и тяло на цикъла

```
for (инициализация; условие; обновяване)
{
    желан сорс код за изпълнение;
}
```

4. **foreach** - служи за обхождане на всички елементи на даден масив, списък или друга колекция от елементи

```
foreach (променлива in колекция)
{
    желан сорс код за изпълнение;
}
```

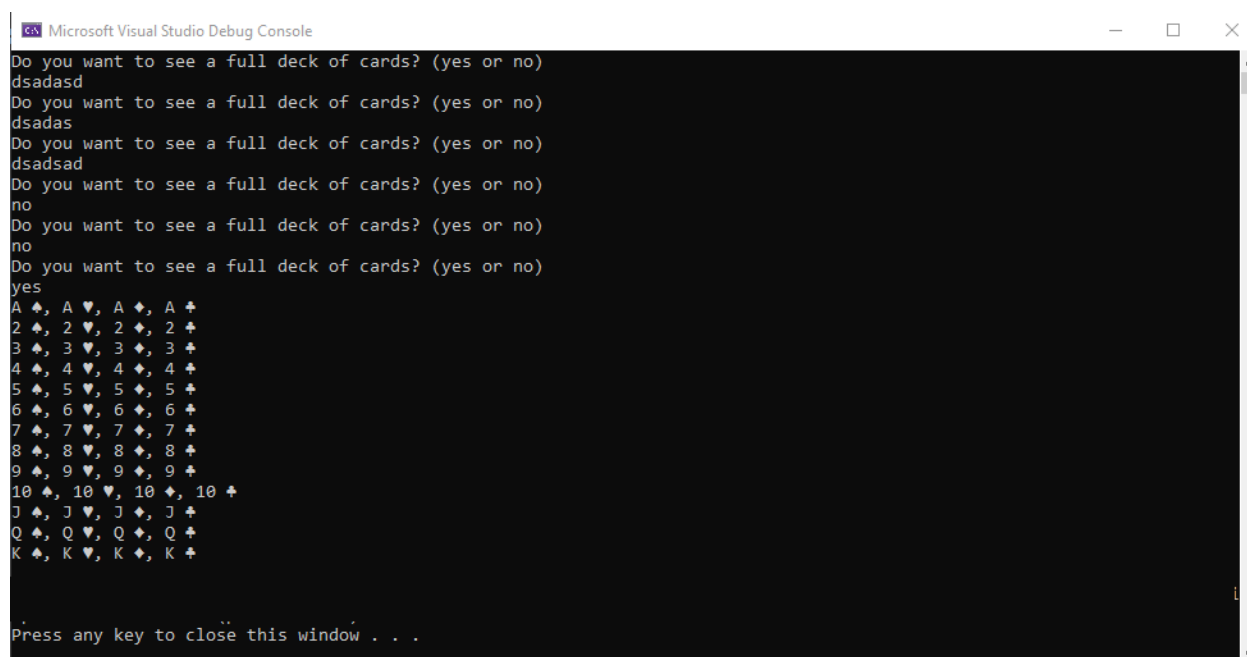
5. вложени цикли - представляват конструкция от няколко цикъла, разположени един в друг. Най-вътрешния цикъл се изпълнява най-много пъти, а най-външният – най-малко

```
for (инициализация; условие; обновяване)
{
    for (инициализация; условие; обновяване)
    {
        желан сорс код за изпълнение;
    }
}
```

## Задача 11: SimpleDeckOfCards

Създайте конзолно приложение, което пита потребителя дали иска да види всички карти от едно тесте. Докато потребителя не даде отговор „yes“, не трябва да спирате да задавате въпроса. При получен отговор „yes“, вашата програма трябва да разпечата тестето в конзолата. Под тесте с карти разбираме 52 карти, които са съставени от 13 карти всяка от която има четири бои (купа, каро, спатия, пика). Програмата трябва да разпечата в конзолата символите на боите.

Желано поведение на програмата:



```
Microsoft Visual Studio Debug Console
Do you want to see a full deck of cards? (yes or no)
dsadasd
Do you want to see a full deck of cards? (yes or no)
dsadasd
Do you want to see a full deck of cards? (yes or no)
dsadasd
Do you want to see a full deck of cards? (yes or no)
no
Do you want to see a full deck of cards? (yes or no)
no
Do you want to see a full deck of cards? (yes or no)
yes
A ♠, A ♥, A ♦, A ♣
2 ♠, 2 ♥, 2 ♦, 2 ♣
3 ♠, 3 ♥, 3 ♦, 3 ♣
4 ♠, 4 ♥, 4 ♦, 4 ♣
5 ♠, 5 ♥, 5 ♦, 5 ♣
6 ♠, 6 ♥, 6 ♦, 6 ♣
7 ♠, 7 ♥, 7 ♦, 7 ♣
8 ♠, 8 ♥, 8 ♦, 8 ♣
9 ♠, 9 ♥, 9 ♦, 9 ♣
10 ♠, 10 ♥, 10 ♦, 10 ♣
J ♠, J ♥, J ♦, J ♣
Q ♠, Q ♥, Q ♦, Q ♣
K ♠, K ♥, K ♦, K ♣
Press any key to close this window . . .
```

## Задача 12: BusTicketLuckyNumber

Компания за транспорт на пътници решава да изненада своите клиенти с лотария. Те имат скромни изисквания:

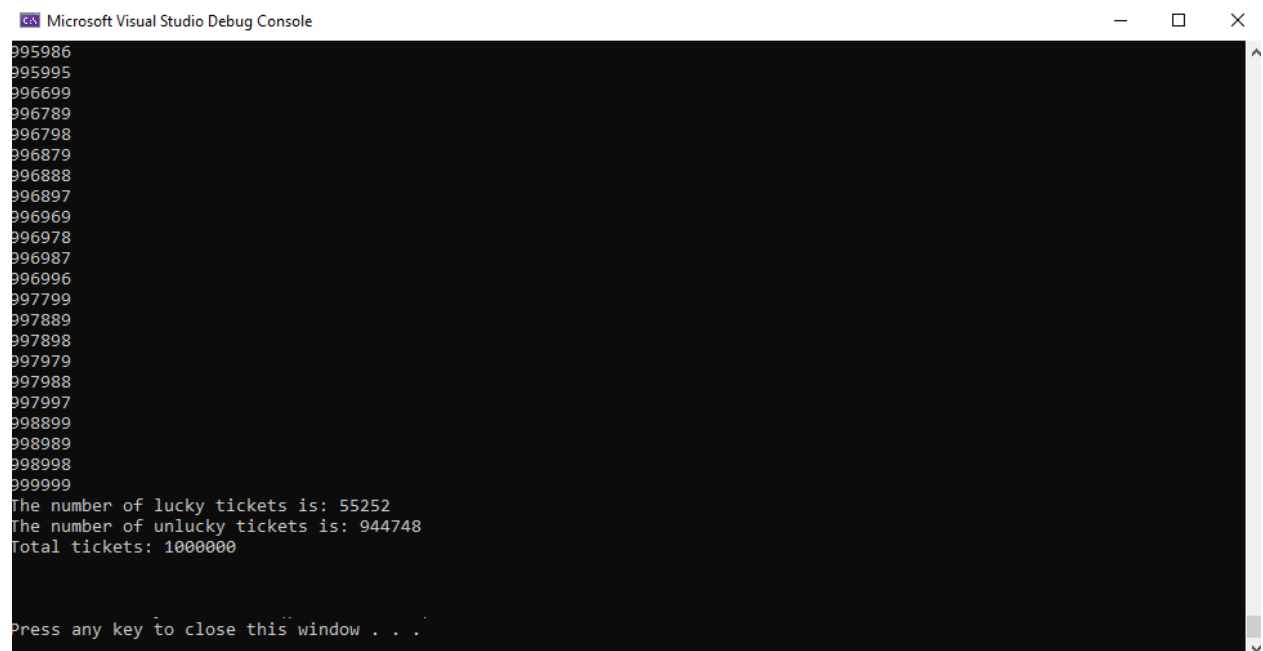
- всеки пътнически билет има уникален шест цифрен номер
- ако сумата на първите три числа от номера са равни на сумата от вторите три, то вашият билет е щастлив
- пример за щастлив пътнически билет 829955 ( $8 + 2 + 9 = 9 + 5 + 5$ )
- всички останали билети се считат за обикновени

Тук идва и вашата задача, направете конзолно приложение, което:

- принтира всички печеливши комбинации в конзолата
- пресмята и принтира колко е броя на всички печеливши билети
- пресмята и принтира колко е броя на всички обикновени билети
- пресмята и принтира колко е броя на всички билети



Желано поведение на програмата:



```
Microsoft Visual Studio Debug Console
995986
995995
996699
996789
996798
996879
996888
996897
996969
996978
996987
996996
997799
997889
997898
997979
997988
997997
998899
998989
998998
999999
The number of lucky tickets is: 55252
The number of unlucky tickets is: 944748
Total tickets: 1000000
Press any key to close this window . . .
```

### Задача 13: MiserPiggy

Заради своята алчност, един ден трите прасенца решили да пресметнат чие име е по-скъпо. Решили, че ще заменят всяка една буква от имената си със стойността и от ASCII таблицата. Сумата от всички букви заменени със стойностите им от ASCII таблицата е равна на стойността на името. Вашата задача е да създадете програма, която извършва сметките и отпечатва в конзолата в низходящ ред резултатите от калкулацията. Имената на прасенцата са Fiffer Pig, Fiddler Pig, Practical Pig. Направете вашата програма да може да работи и с други имена като ги взима като вход от конзолата.

Проучете какво представлява ASCII таблицата.

Вход	Изход
Fiffer Pig	The name-wealthiest pig is Practical Pig (1235) followed by Fiddler Pig (1018) and last is Fiffer Pig (914).
Fiddler Pig	
Practical Pig	

Желано поведение на програмата:

```
Microsoft Visual Studio Debug Console
Hey, piggies! Enter your names:
Piggy 1: Fiffer Pig
Piggy 2: Fiddler Pig
Piggy 3: Practical Pig
The name-wealthiest pig is Practical Pig (1235) followed by Fiddler Pig (1018) and last is Fiffer Pig (914).

Press any key to close this window . . .
```

## 11. Масиви

Масивите представляват съвкупности от променливи, които наричаме елементи. Елементите на масивите в C# са номерирани с числата 0, 1, 2, ... N-1. Тези номера на елементи се наричат индекси. Броят елементи в даден масив се нарича дължина на масива. Всички елементи на даден масив са от един и същи тип, независимо дали е примитивен или референтен. Това ни помага да представим група от еднородни елементи като подредена свързана последователност и да ги обработваме като едно цяло. Масивите могат да бъдат от различни размерности, като най-често използвани са едномерните и двумерните масиви. Едномерните масиви се наричат още вектори, а двумерните – матрици. В C# масивите имат фиксирана дължина, която се указва при инициализирането им и определя броя на елементите им. След като веднъж е зададена дължината на масив при неговото създаване, след това не е възможно да се променя. Основни начини за декларация на масив:

```
int[] myArray = new int[6];
```

```
int[] myArray = { 1, 2, 3, 4, 5, 6 };
```

За да бъдат разпечатани стойностите на един масив, то той трябва да бъде обходен чрез цикъл и на всяка една итерация отпечатваме текущия елемент.

### Задача 14: WorkingWithSomeIntegers

Направете програма, която генерира 100 случайни числа в диапазона [0, 132] и ги разпечатва в конзолата. След като е разпечатала случайните числа, програмата трябва да даде право на потребителя да избере информацията, която му е нужна по формата на команда. Командите за улеснение ще бъдат букви, зад всяка буква стои съответно действието:

- отпечатва всички числа с четни индекси

- b) отпечатва всички нечетни числа, които имат същевременно и нечетни индекси
- c) отпечатва всички числа, които се делят на 3 без остатък
- d) отпечатва всички числа, които се делят на 7 и имат остатък 1
- e) отпечатва всички числа, които се намират в интервала [26, 100]
- f) отпечатва всички числа, които не се намират в интервала [26, 100]

Проучете как и защо се използва класа Random.

Желано поведение на програмата (примера е от команда с буква „b“):

```

Microsoft Visual Studio Debug Console
Choose what you want to do with them (type in the letter):
a) Display the numbers that have even indexes.
b) Display the numbers that are odd and have odd indexes.
c) Display the numbers that are divided by 3 without a remainder.
d) Display the numbers that are divided by 7 and have 1 as remainder.
e) Display the numbers that are in the interval between 26 and 100.
f) Display the numbers that are not in the interval between 26 and 100.
-----
b
-----
57 is odd and has index 5.
39 is odd and has index 7.
121 is odd and has index 13.
25 is odd and has index 15.
79 is odd and has index 19.
93 is odd and has index 31.
107 is odd and has index 35.
79 is odd and has index 37.
75 is odd and has index 41.
17 is odd and has index 47.
31 is odd and has index 55.
27 is odd and has index 61.
129 is odd and has index 67.
105 is odd and has index 69.
83 is odd and has index 81.
77 is odd and has index 83.
111 is odd and has index 85.
17 is odd and has index 95.
41 is odd and has index 97.

```

### Задача 15: WorkingWithSomeLetters

Направете програма, която генерира редица от 100 случайни символа (български букви). Да се отпечатат тези от тях, които са:

- широка гласна (а, е, о)
- тясна гласна (и, у, ъ)
- съставна гласна (ю, я)
- беззвучна съгласна (п, ф, к, т, ш, с, х, ц)
- сонорна съгласна (л, м, н, р)
- звучна съгласна (б, в, г, д, ж, з, ч)

В отпечатването на информацията трябва да се съдържа и на кое място в редицата се намира буквата.

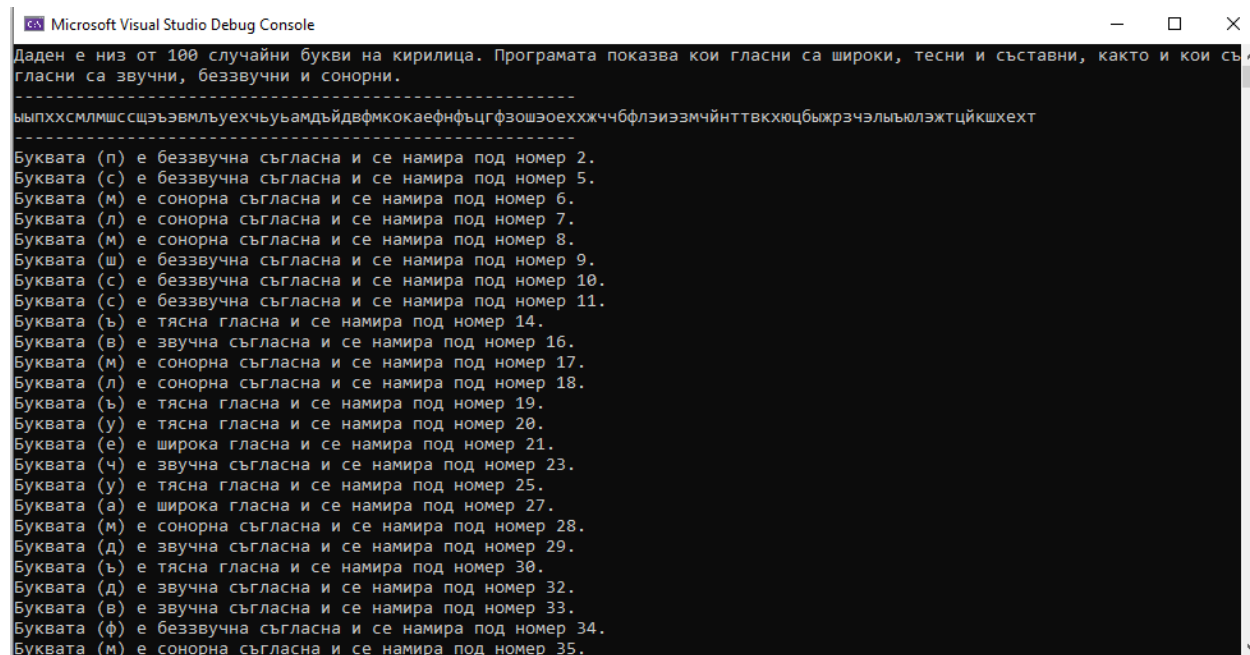
Пример за редица от 100 случайни букви на кирилица:

“уеишщкскдзцъяаожгтнвмчюйъэфхпрлбуеишщкскдзцъяаожгтнвмчюйъэфхпрлбуеишщкск  
дзцъяаожгтнвмчюйъэфхпрлбуеишщк”

За да работи с български букви, вашата програма трябва да съдържа програмен код за енкодинг, който изглежда по следния начин:

Console.OutputEncoding = Encoding.UTF8;

Желано поведение на програмата:



```
Microsoft Visual Studio Debug Console
Даден е низ от 100 случайни букви на кирилица. Програмата показва кои гласни са широки, тесни и съставни, както и кои съ
гласни са звучни, беззвучни и сонорни.
-----
ыпххсмлмшссщэъэмльуехчъуъамдъдвфмкоаефнфъцгфзошэоеххжчбфлэизмчйнттвкхюцбыжрзчэлыьолэжтцйкшхехт
-----
Буквата (п) е беззвучна съгласна и се намира под номер 2.
Буквата (с) е беззвучна съгласна и се намира под номер 5.
Буквата (м) е сонорна съгласна и се намира под номер 6.
Буквата (л) е сонорна съгласна и се намира под номер 7.
Буквата (м) е сонорна съгласна и се намира под номер 8.
Буквата (ш) е беззвучна съгласна и се намира под номер 9.
Буквата (с) е беззвучна съгласна и се намира под номер 10.
Буквата (с) е беззвучна съгласна и се намира под номер 11.
Буквата (ъ) е тясна гласна и се намира под номер 14.
Буквата (в) е звучна съгласна и се намира под номер 16.
Буквата (м) е сонорна съгласна и се намира под номер 17.
Буквата (л) е сонорна съгласна и се намира под номер 18.
Буквата (ъ) е тясна гласна и се намира под номер 19.
Буквата (у) е тясна гласна и се намира под номер 20.
Буквата (е) е широка гласна и се намира под номер 21.
Буквата (ч) е звучна съгласна и се намира под номер 23.
Буквата (у) е тясна гласна и се намира под номер 25.
Буквата (а) е широка гласна и се намира под номер 27.
Буквата (м) е сонорна съгласна и се намира под номер 28.
Буквата (д) е звучна съгласна и се намира под номер 29.
Буквата (ъ) е тясна гласна и се намира под номер 30.
Буквата (д) е звучна съгласна и се намира под номер 32.
Буквата (в) е звучна съгласна и се намира под номер 33.
Буквата (ф) е беззвучна съгласна и се намира под номер 34.
Буквата (м) е сонорна съгласна и се намира под номер 35.
```

## 12. Методи

В ежедневието ни, при решаването на даден проблем, особено, ако е по-сложен, прилагаме принципа на древните римляни "Разделяй и владей". Съгласно този принцип, проблемът, който трябва да решим, се разделя на множество по-малки подпроблеми. Самостоятелно разгледани, те са по-ясно дефинирани и по-лесно решими, в сравнение с търсенето на решение на изходния проблем като едно цяло. Накрая, от решенията на всички подпроблеми, създаваме решението на цялостния проблем. По същата аналогия, когато пишем дадена програма, целта ни е с нея да решим конкретна задача. За да го направим ефективно и да улесним работата си, прилагаме принципа "Разделяй и владей". Разбиваме поставената ни задача на подзадачи, разработваме решения на тези подзадачи и накрая ги "сглобяваме" в една програма. Решенията на тези подзадачи наричаме подпрограми. Метод (method) е съставна част от програмата, която решава даден проблем, може да приема параметри и да връща стойност. Структурата на един метод изглежда по следния начин:

```
static void ShowTheNameInConsole(string name)
{
    Console.WriteLine(name);
}
```

### Задача 16: GetMax

Създайте метод на име GetMax, който взима като параметри три числа и връща стойността на най-голямото от тях. Създайте програма, която взима като вход от конзолата три числа и използва метода GetMax, след това отпечатва в конзолата резултата от метода.

Вход	Изход
2	The biggest number is: 4
3	
4	

Желано поведение на програмата:

```
Microsoft Visual Studio Debug Console
Enter 3 numbers.
Enter the first number: 2
Enter the second number: 3
Enter the third number: 4
The biggest number is: 4

Press any key to close this window . . .
```

### Задача 17: GeometryCalculator

Създайте програма, която може да пресмята лицата на две различни геометрични фигури:

- Триъгълник
- Квадрат

Третото условие е да може да преобразува градуси в радиани. Формулата за преобразуване на градуси в радиани е ( $\text{Rad} = \pi/180 * \text{градусите}$ ).

В началото програмата трябва да попита потребителя коя функция иска да използва и да показва специфично меню спрямо неговият избор. Решението трябва да бъде реализирано чрез методи.

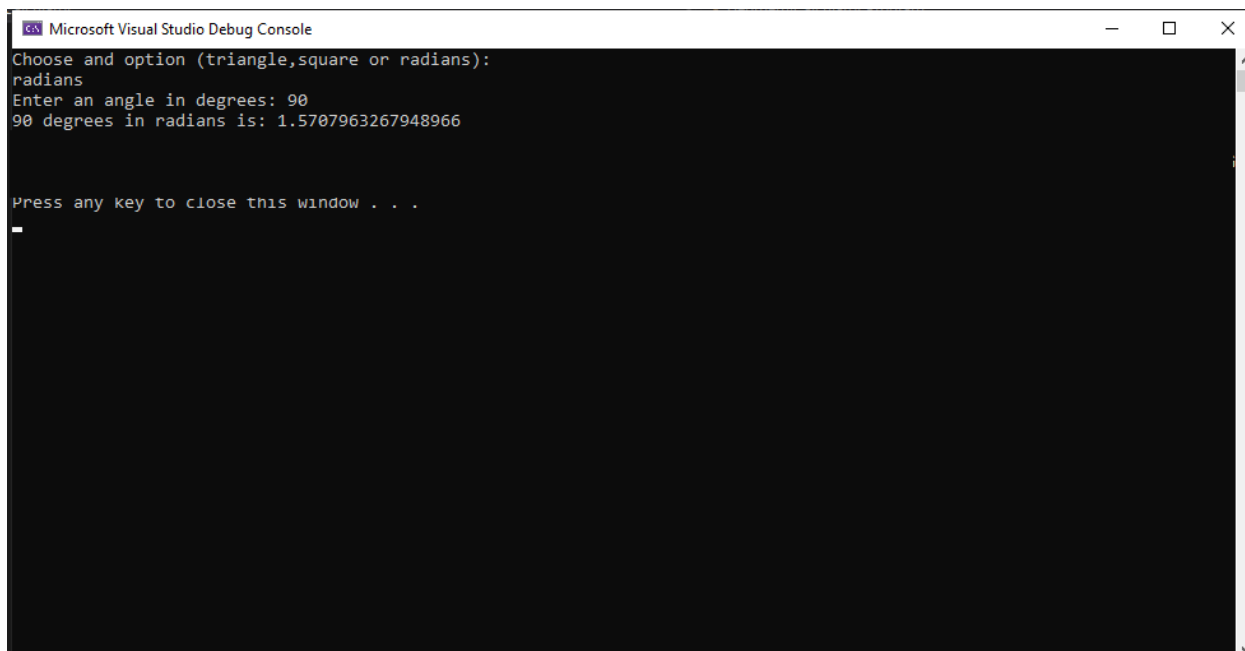
Желано поведение на програмата:

```
Microsoft Visual Studio Debug Console
Choose and option (triangle,square or radians):
triangle
Enter a side: 5
Enter the height to that side: 12
The area of this triangle is: 30

Press any key to close this window . . .
```

```
Microsoft Visual Studio Debug Console
Choose and option (triangle,square or radians):
square
Enter a side: 5
The area of this square is: 25

Press any key to close this window . . .
```



```
Microsoft Visual Studio Debug Console
Choose and option (triangle,square or radians):
radians
Enter an angle in degrees: 90
90 degrees in radians is: 1.5707963267948966

Press any key to close this window . . .
```

### 13. Основни методи при работа със символни низове

В модерните приложения постоянно се налага да се извършват манипулации върху символни низове (текстове). В тази тема ще се запознаем с някои от основните методи на класа String:

- String.ToLower(); - всички букви стават малки
- String.ToUpper(); - всички букви стават главни
- String.IndexOf (подниз); - връща позицията, на която се намира подниза
- String.Substring(начален индекс, дължина след началото); - връща подниз, който започва от началния индекс и продължава до броя на дължината след началото
- String.Split(символ за разделяне); - връща масив от поднизове, които са разделени по символа за разделяне
- String.Replace(подниз за замяна, желан нов подниз на негово място); - заменя стар подниз с нов
- String.Trim(); - премахва празните места в началото и края на низа

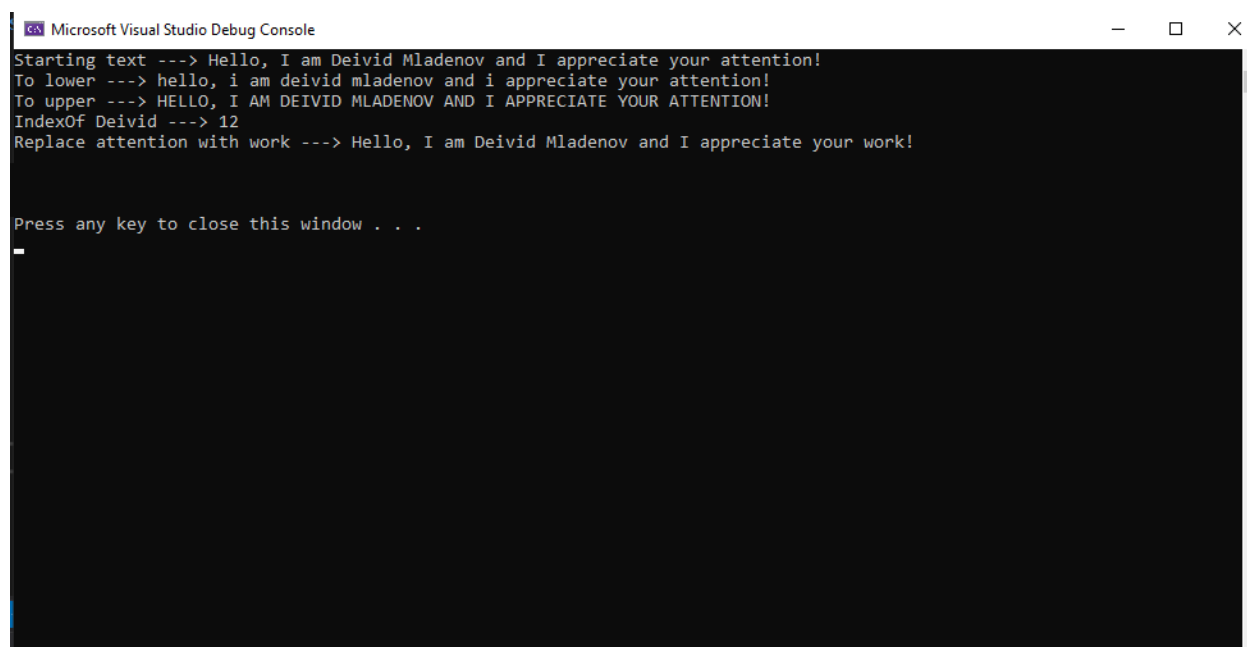
## Задача 18: StringManipulations

`string` text = "Hello, I am Deivid Mladenov and I appreciate your attention!"

Създайте програма, която извършва следните манипулации върху даденият символен низ и ги разпечатва в конзолата:

- началната стойност на низа
- превръща стойността на низа в малки букви
- превръща стойността на низа в главни букви
- извежда индекса на подниза „Deivid“
- замества „attention“ с „work“

Желано поведение на програмата:



```
Microsoft Visual Studio Debug Console
Starting text ---> Hello, I am Deivid Mladenov and I appreciate your attention!
To lower ---> hello, i am deivid mladenov and i appreciate your attention!
To upper ---> HELLO, I AM DEIVID MLADENOV AND I APPRECIATE YOUR ATTENTION!
IndexOf Deivid ---> 12
Replace attention with work ---> Hello, I am Deivid Mladenov and I appreciate your work!

Press any key to close this window . . .
```

## 14. Обработка на изключения

Изключение (exсeption) в програмирането, в общия случай, представлява уведомление за дадено събитие, нарушаващо нормалната работа на една програма. Изключенията дават възможност необичайните събития да бъдат обработвани и програмата да реагира на тях по някакъв начин. Когато възникне изключение, конкретното състояние на програмата се запазва и се търси обработчик на изключението (exсeption handler).

Изключенията се предизвикват или "хвърлят" (throw an exсeption) от програмен код, който трябва да сигнализира на изпълняващата се програма за грешка или необичайна ситуация. Например ако се опитваме да въведем символен низ, но



програмата очаква число, кодът, който чете входа от конзолата, ще установи това и ще хвърли изключение с подходящо съобщение за грешка.

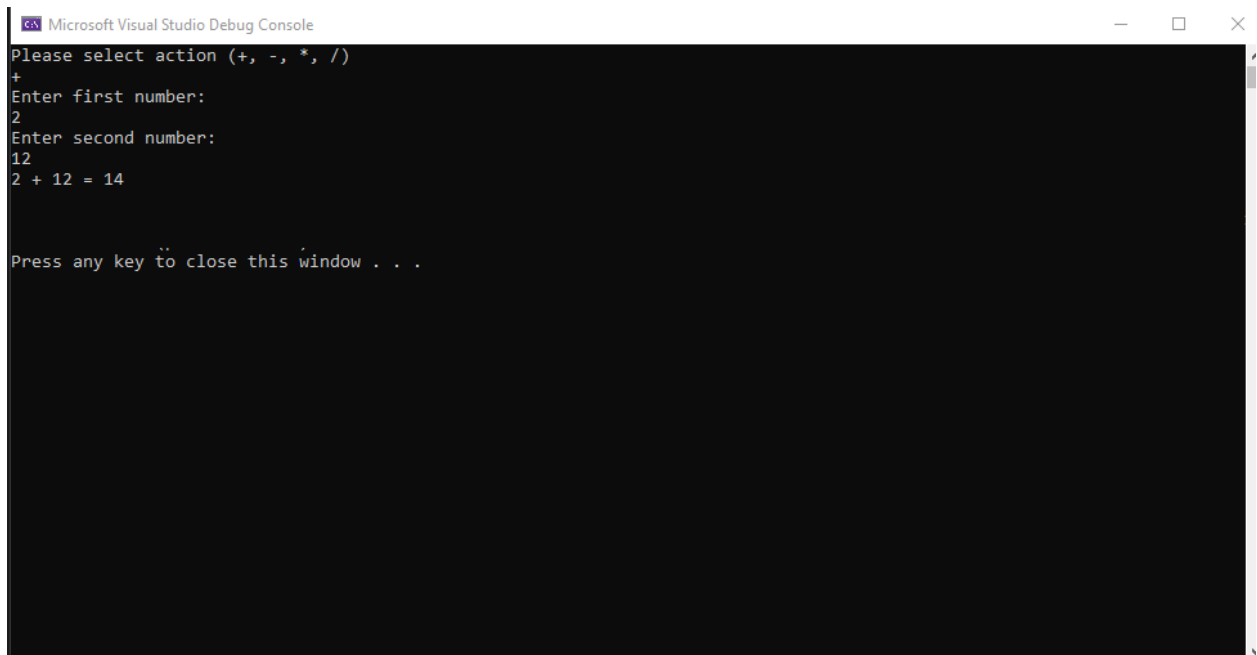
В C# има заложен подход, чрез който се прихващат изключенията. Този подход е познат като използване на try-catch блокове. Основна try-catch структура:

```
try
{
    код, който очаквате, че може да хвърли изключение
}
catch (Exception ex)
{
    код за обработка на изключението
}
```

### Задача 19: JustCalculator

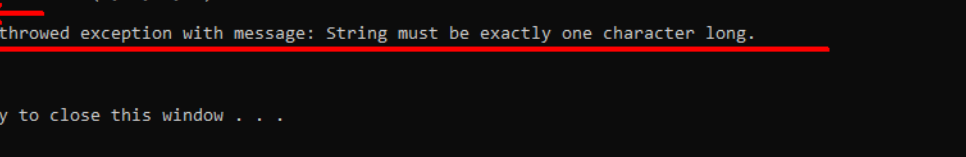
Създайте програма, която извършва базовите операции на калкулатор (+, -, \*, /) и изписва резултатите от действията в конзолата. Подсигурете, че вашата програма обработва изключенията и изписва в конзолата точното съобщение, което конкретното изключение съдържа.

Желано поведение на програмата:



```
Microsoft Visual Studio Debug Console
Please select action (+, -, *, /)
+
Enter first number:
2
Enter second number:
12
2 + 12 = 14

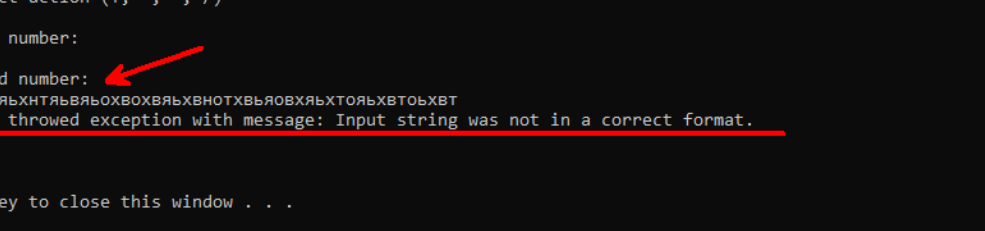
Press any key to close this window . . .
```



The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio Debug Console". The console output is as follows:

```
Please select action (+, -, *, /)
32131321
The program threw exception with message: String must be exactly one character long.
```

A red arrow points to the input "32131321". The exception message "The program threw exception with message: String must be exactly one character long." is underlined in red. At the bottom of the console, it says "Press any key to close this window . . .".

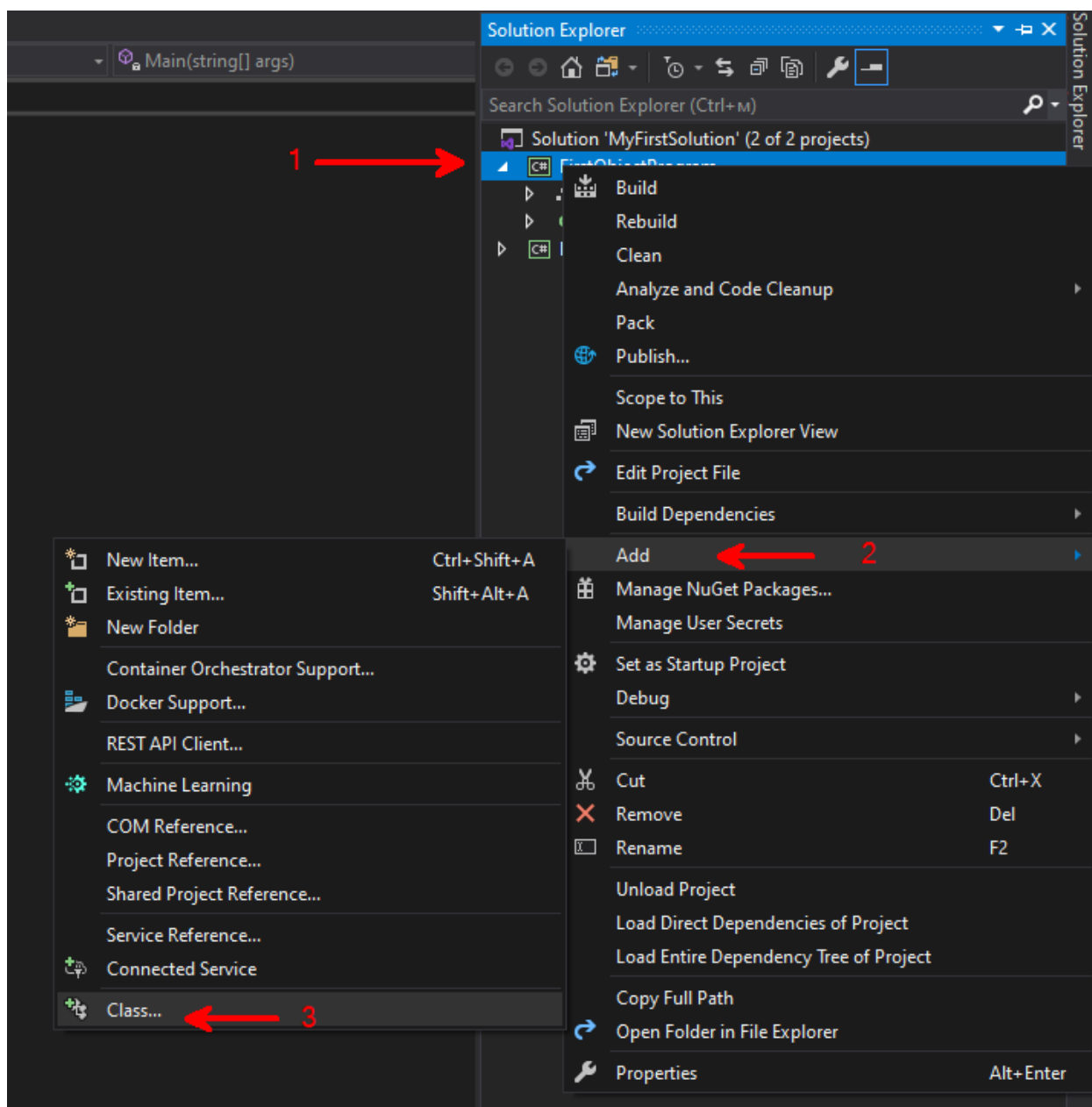


```
Microsoft Visual Studio Debug Console
Please select action (+, -, *, /)
*
Enter first number:
2
Enter second number:
аЪяРоПнаЪПХОяЪХНТяЪВяЪОХВОХВяЪХВНОТХВЪяОВХяЪХТОяЪХВТОЪХВТ
The program threw exception with message: Input string was not in a correct format.
Press any key to close this window . . .
```

## 15. Създаване и използване на обекти

Обектно-ориентираното програмиране е модел на програмиране, който използва обекти и техните взаимодействия за изграждането на компютърни програми. По този начин се постига лесен за разбиране, опростен модел на предметната област, който дава възможност на програмиста интуитивно (чрез проста логика) да решава много от задачите, които възникват в реалния свят. Класът дефинира абстрактните характеристики на даден обект. Той е план или шаблон, чрез който се описва природата на нещо (някакъв обект). Класовете са градивните елементи на ООП и са неразделно свързани с обектите. Нещо повече, всеки обект е представител на точно един клас.

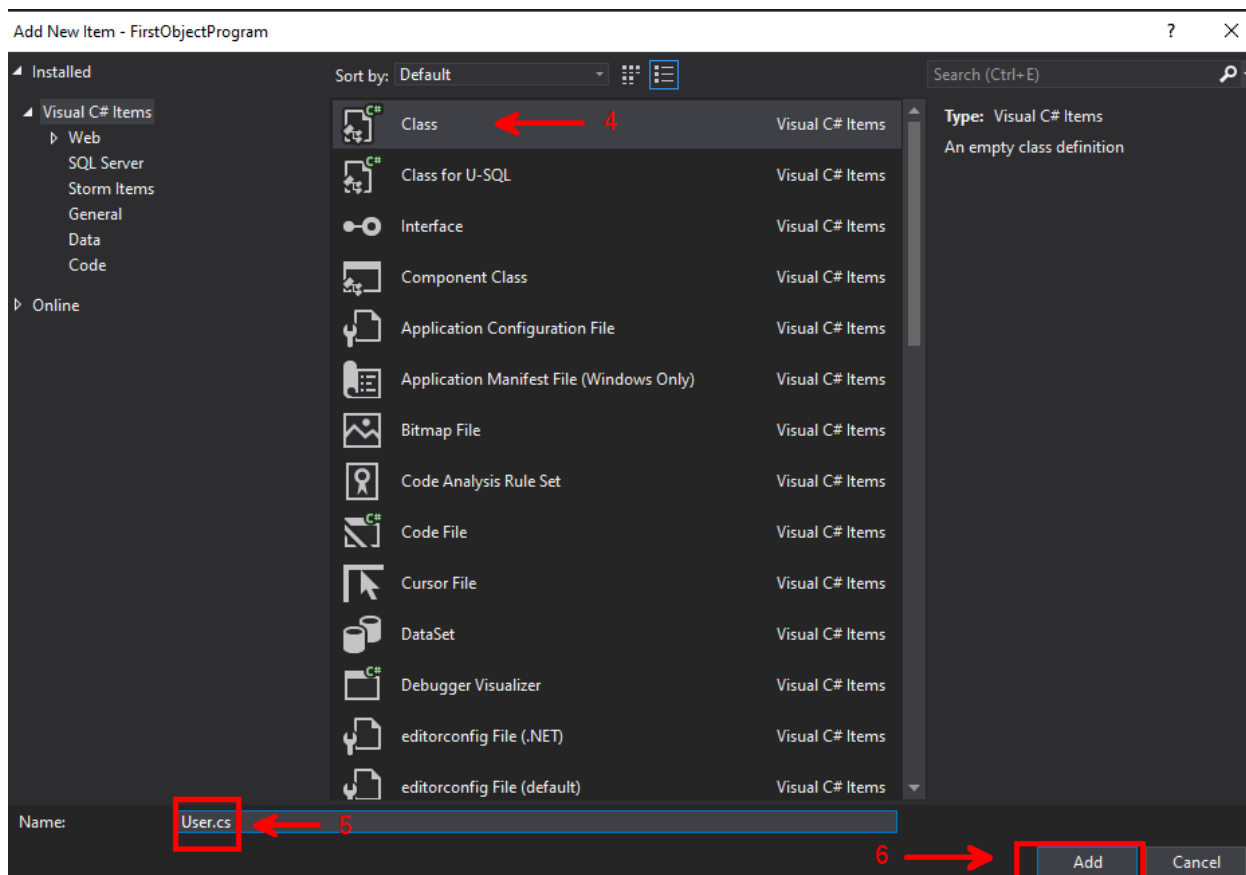
Ще дадем базов пример за това как може да създадем и използваме обект в нашата програма. За примера ще създадем едно нормално конзолно приложение и към него ще добавим обект, който ще именуваме User.



1 – Десен бутон върху конзолното приложение

2 – Избираме Add

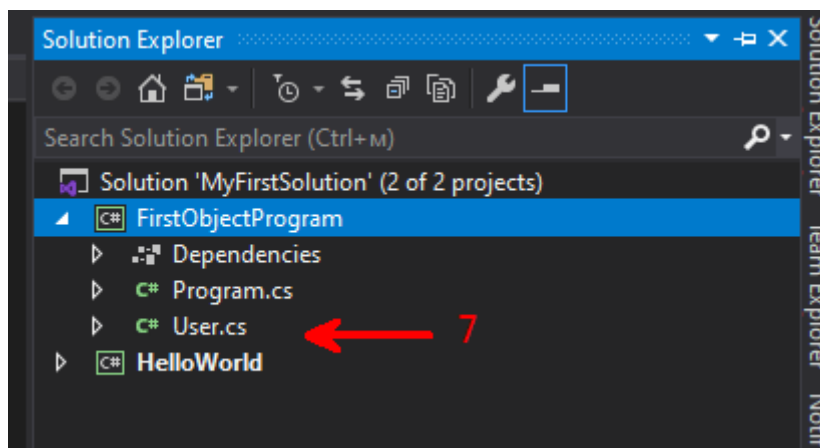
3 – Class



4 – Подсигуряваме се че сме избрали Class

5 – Именуваме с разбираемо име, в нашият случай User

6 – Натискаме Add



7 – Виждаме, че нашият клас е успешно създаден

```

1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4
5  namespace FirstObjectProgram
6  {
7      public class User
8      {
9      }
10 }
11

```

0 references ← 8

8 – Задаваме нивото на достъп (access modifier) public за момента, в бъдеще ако учите ООП ще се запознаете с тях в детайл

```

1  using System;
2  using System.Collections.Generic;
3  using System.Text;
4
5  namespace FirstObjectProgram
6  {
7      public class User
8      {
9          public string Email { get; set; }
10
11         public string FirstName { get; set; }
12
13         public string LastName { get; set; }
14
15         public int Age { get; set; }
16     }
17 }

```

0 references

0 references ← 9

0 references

0 references

0 references

9 – Дефинираме свойствата (properties) на обекта

```

1  using System;
2
3  namespace FirstObjectProgram
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              User user = new User(); ← 10
10             user.Email = "deividmladenov007@gmail.com";
11             user.FirstName = "Deivid";
12             user.LastName = "Mladenov";
13             user.Age = 22;
14
15             Console.WriteLine($"{user.FirstName + ' ' + user.LastName} is {user.Age} years old and his email is {user.Email}"); ← 12
16         }
17     }
18 }

```

10 – Инстанцираме нашият обект User в Program.ts файла на конзолното ни приложение

11 – Попълваме стойности на неговите свойства

12 – Разпечатваме стойностите на обекта

```

5  namespace FirstObjectProgram
6  {
7      public class User
8      {
9          public User(string email, string firstName, string lastName, int age) ← 13
10         {
11             this.Email = email;
12             this.FirstName = firstName;
13             this.LastName = lastName;
14             this.Age = age;
15         }
16
17         public string Email { get; set; }
18
19         public string FirstName { get; set; }
20
21         public string LastName { get; set; }
22
23         public int Age { get; set; }
24     }
25 }

```

13 – На стъпка 10 създадохме инстанция на обекта User, но той беше празен и се наложи на стъпка 11 да попълним ръчно стойностите на свойствата му. Това е много несигурен подход, затова може да използваме конструктор (constructor), в който да дефинираме нужните стойности на свойствата още при инстанцирането

```

1  using System;
2
3  namespace FirstObjectProgram
4  {
5      0 references
6      class Program
7      {
8          0 references
9          static void Main(string[] args)
10         {
11             User user = new User(); ← 14
12             user.Email = "7deividmladenov007@gmail.com";
13             user.FirstName = "Deivid";
14             user.LastName = "Mladenov";
15             user.Age = 22;
16             Console.WriteLine($"{user.FirstName + ' ' + user.LastName} is {user.Age} years old and his email is {user.Email}");
17         }
18     }

```

14 – След създаването на конструктор с параметри, компилаторът дава съобщение за грешка и ни подсказва, че трябва да въведем нужните стойности, за да бъде създадена инстанция

```

1  using System;
2
3  namespace FirstObjectProgram
4  {
5      0 references
6      class Program
7      {
8          0 references
9          static void Main(string[] args)
10         {
11             User user = new User("7deividmladenov007@gmail.com", "Deivid", "Mladenov", 22); ← 15
12             Console.WriteLine($"{user.FirstName + ' ' + user.LastName} is {user.Age} years old and his email is {user.Email}");
13         }
14     }

```

15 – Въвеждаме стойностите в конструктора

```

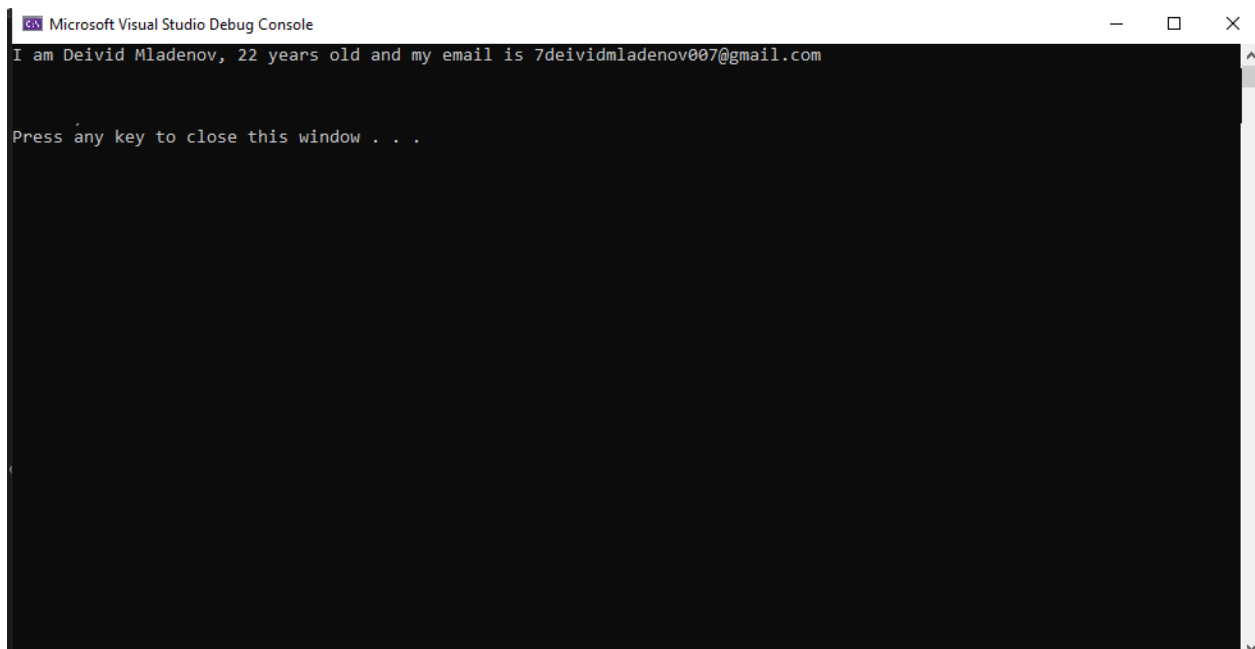
7  3 references
8  public class User
9  {
10     1 reference
11     public User(string email, string firstName, string lastName, int age)
12     {
13         this.Email = email;
14         this.FirstName = firstName;
15         this.LastName = lastName;
16         this.Age = age;
17     }
18
19     2 references
20     public string Email { get; set; }
21
22     2 references
23     public string FirstName { get; set; }
24
25     2 references
26     public string LastName { get; set; }
27
28     2 references
29     public int Age { get; set; }
30
31     1 reference
32     public void IntroduceMyself()
33     {
34         Console.WriteLine($"I am {FirstName + ' ' + LastName}, {Age} years old and my email is {Email}"); ← 16
35     }
36 }

```

16 – Класовете ни дават право да създаваме методи вътре в тях, затова в нашият клас User ще създадем метод, който го представя като изписва данните в конзолата. Ще именуваме нашият метод с говорещо за извършваното от него действие име, а именно IntroduceMyself

```
1  using System;
2
3  namespace FirstObjectProgram
4  {
5      0 references
6      class Program
7      {
8          0 references
9          static void Main(string[] args)
10         {
11             User user = new User("7deividmladenov007@gmail.com", "Deivid", "Mladenov", 22);
12             user.IntroduceMyself();
13         }
14     }
```

17 – Извикваме метода IntroduceMyself през създадената инстанция в Program.cs файла и виждаме резултата от неговото изпълнение след стартиране



```
Microsoft Visual Studio Debug Console
I am Deivid Mladenov, 22 years old and my email is 7deividmladenov007@gmail.com
Press any key to close this window . . .
```

## Задача 20: UserWithActions

Като използвате демото за създаване на класове разширете решението, като добавите нови функционалности към него:

1. Добавете ново свойство на User класа наречено City
2. Добавете нов метод AmIAAdult, който изписва в конзолата дали User е пълнолетен или не спрямо Age, който е въвел
3. Добавете нов метод WhereAmIFrom, който изписва в конзолата от къде е User спрямо City
4. Всички свойства на User трябва да се попълват от конзолата
5. Добавете меню, от което може да се избира кой метод да се активира

Желано поведение на програмата:



```
Microsoft Visual Studio Debug Console

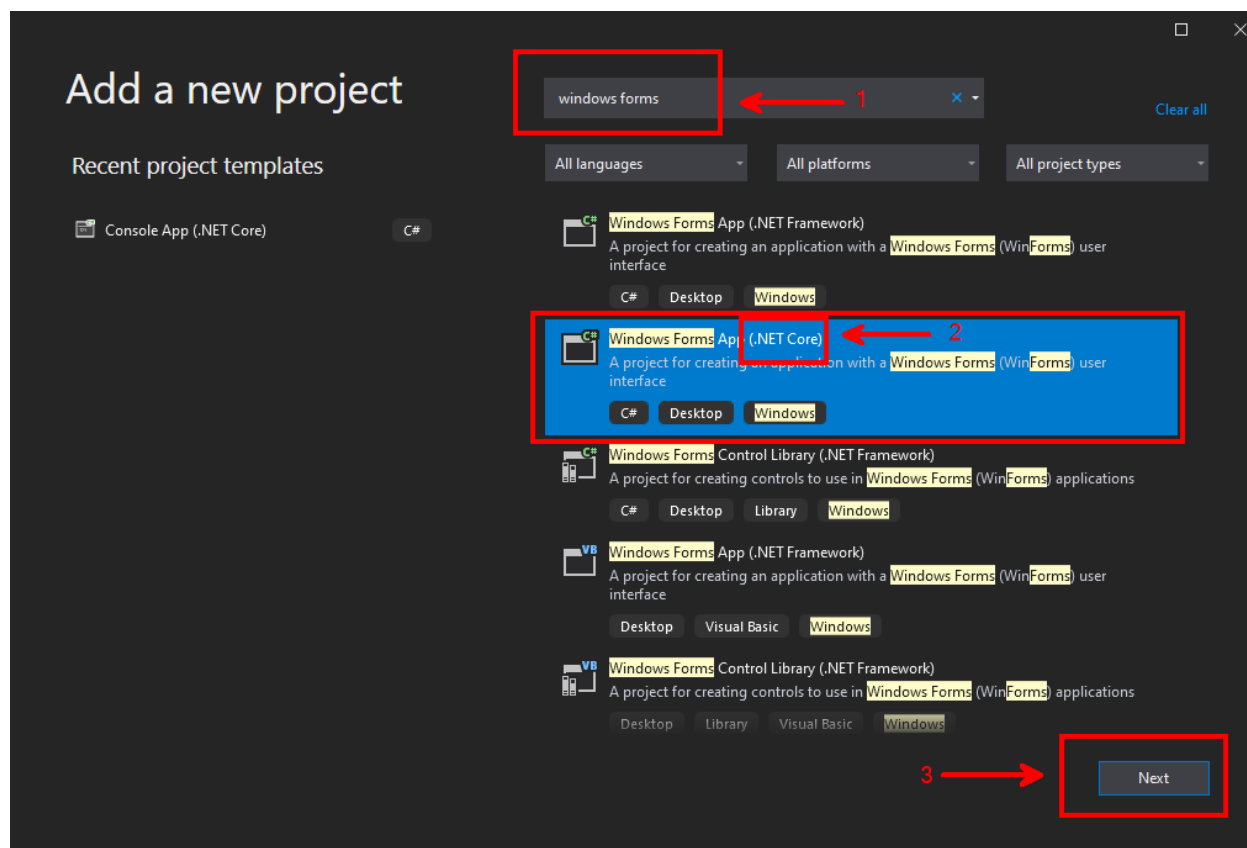
Enter email:
7deividmladenov007@gmail.com
Enter first name:
Deivid
Enter last name:
Mladneov
Enter age:
22
Enter city:
Plovdiv
Select option:
1) Introduce user
2) Is user adult or not
3) Where is user from
3
I am from Plovdiv

Press any key to close this window . . .
```

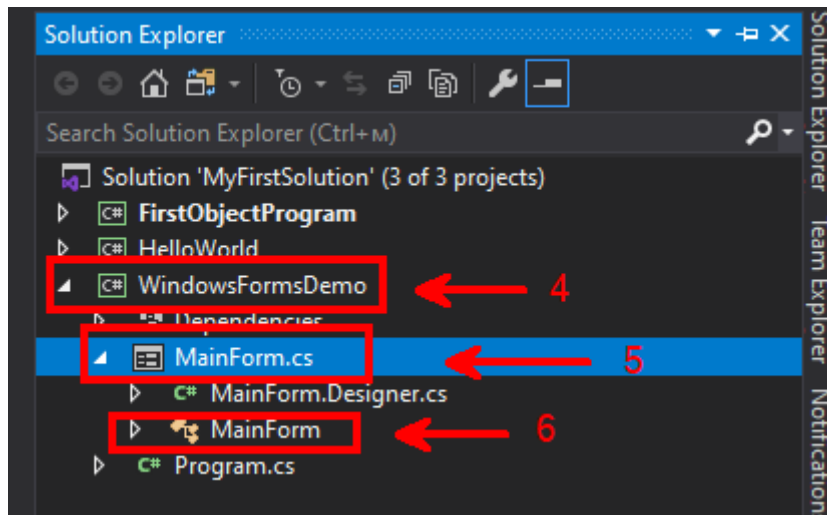
## 16. Приложения с графичен потребителски интерфейс

Освен така добре познатите ни вече конзолни приложения, ще разгледаме и приложения с графичен потребителски интерфейс и основни начини за работа с тях.

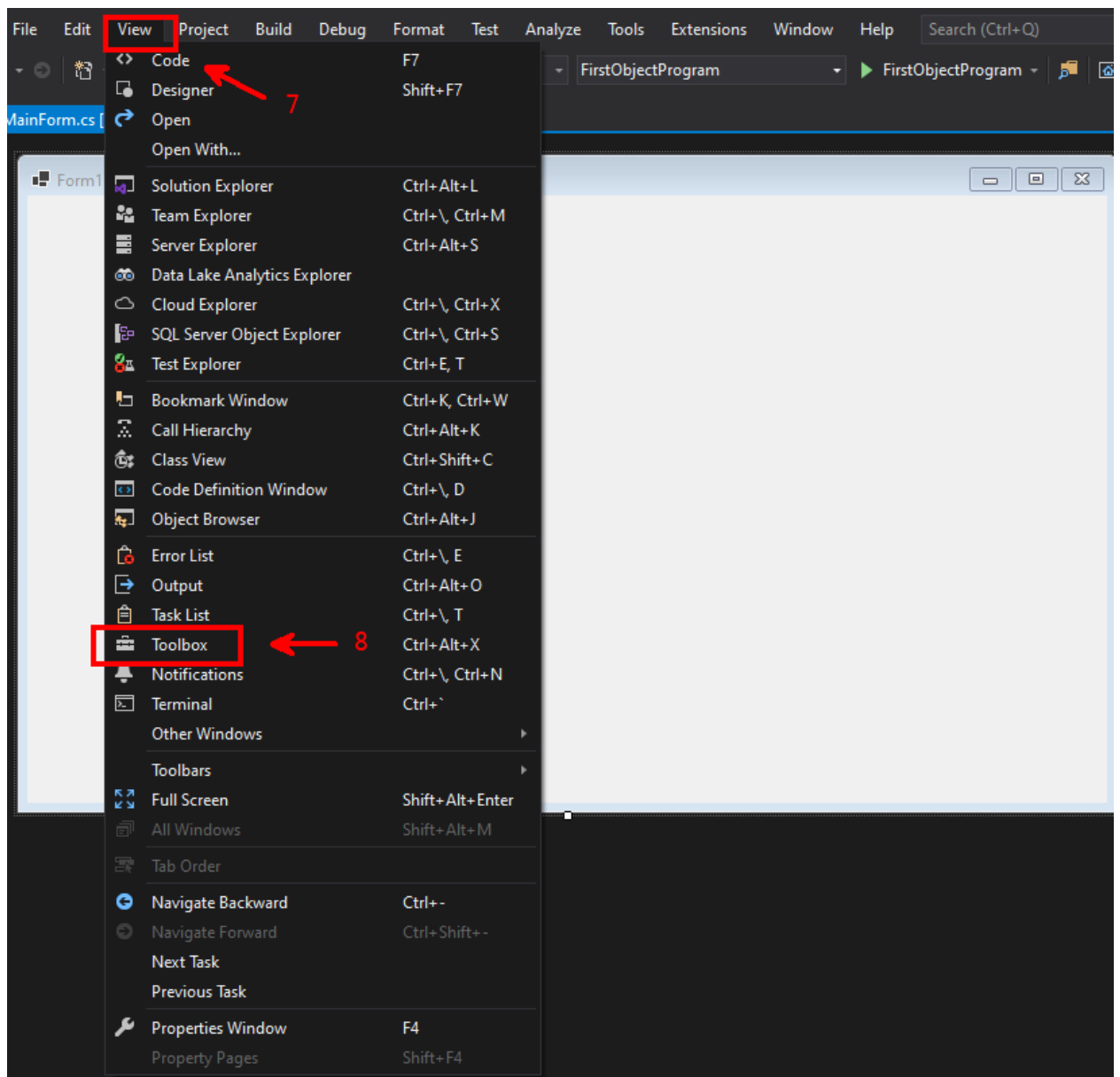
Добавянето на този вид приложение става по аналогичен на конзолните приложения начин.



- 1- Пишем в търсачката windows forms
- 2- Селектираме Windows Forms App (.NET Core)
- 3- Натискаме бутона Next

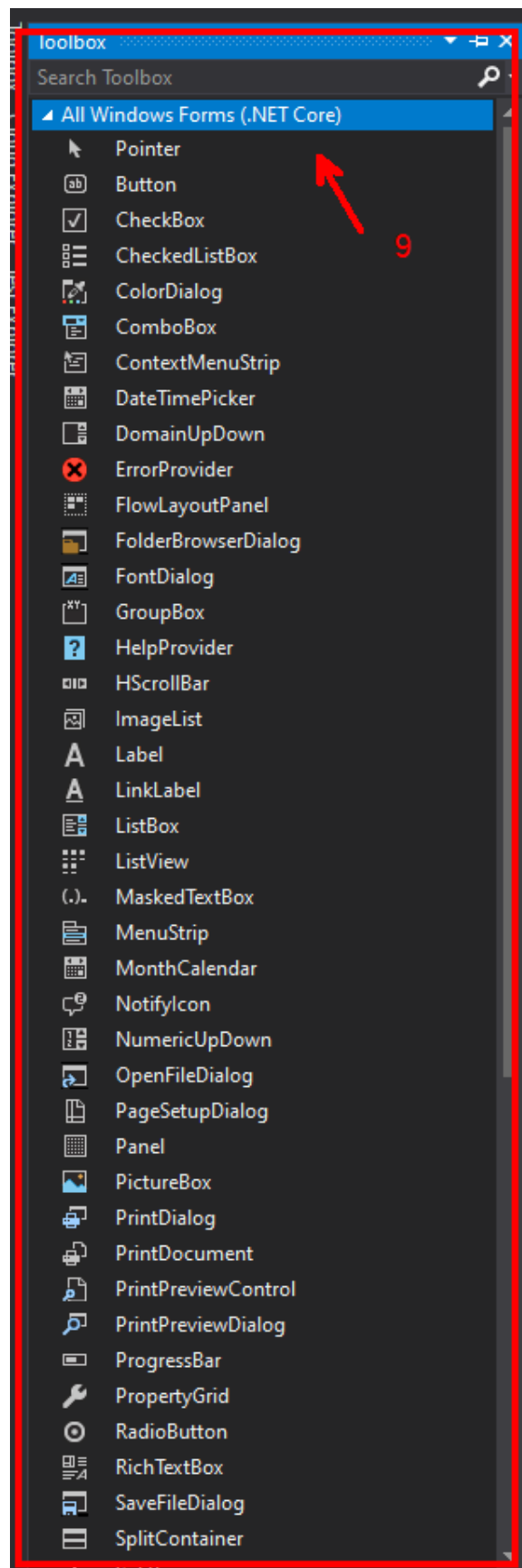


- 4- След като го създадем виждаме, че има малко различия от досегашните конзолни приложение и по-специално в архитектурата
- 5- Когато натиснем върху MainForm, както съм именувал моята форма в момента, се отваря дизайнер, който ни помага да наредим така нареченият графичен потребителски интерфейс
- 6- MainForm файла съдържа имплементацията на методите към текущата форма

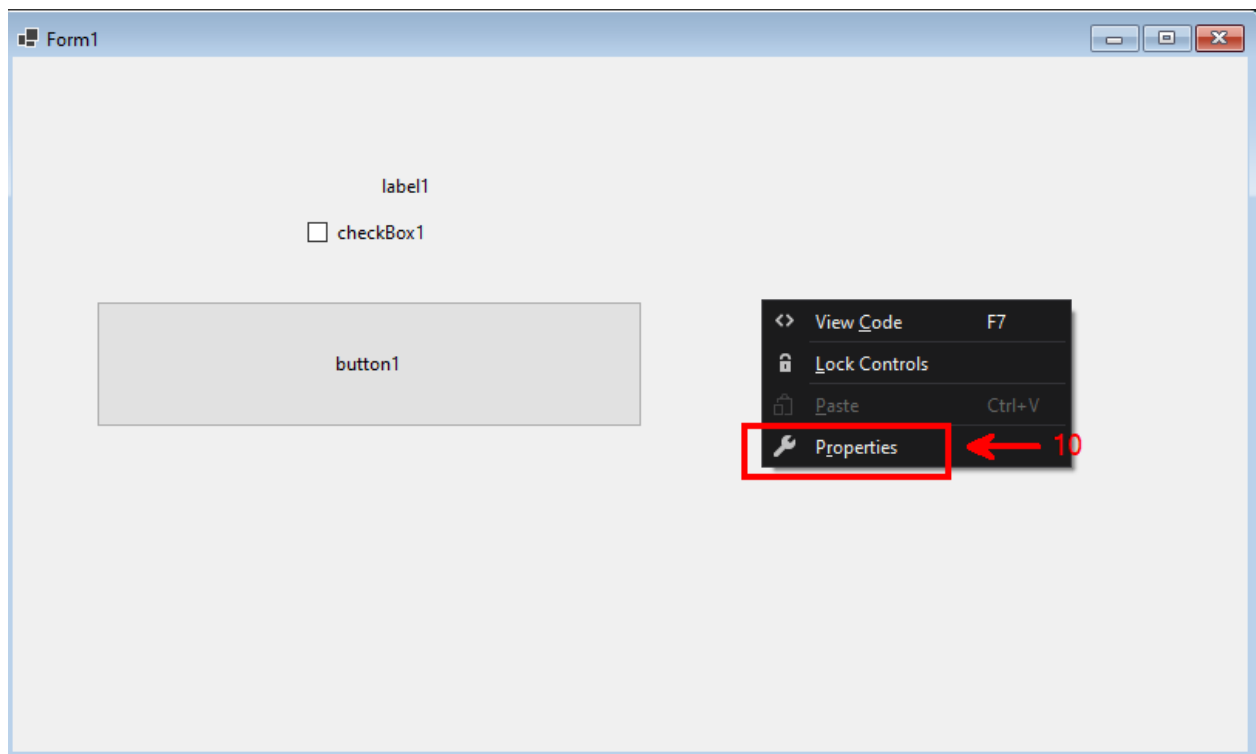


7- Отиваме на View

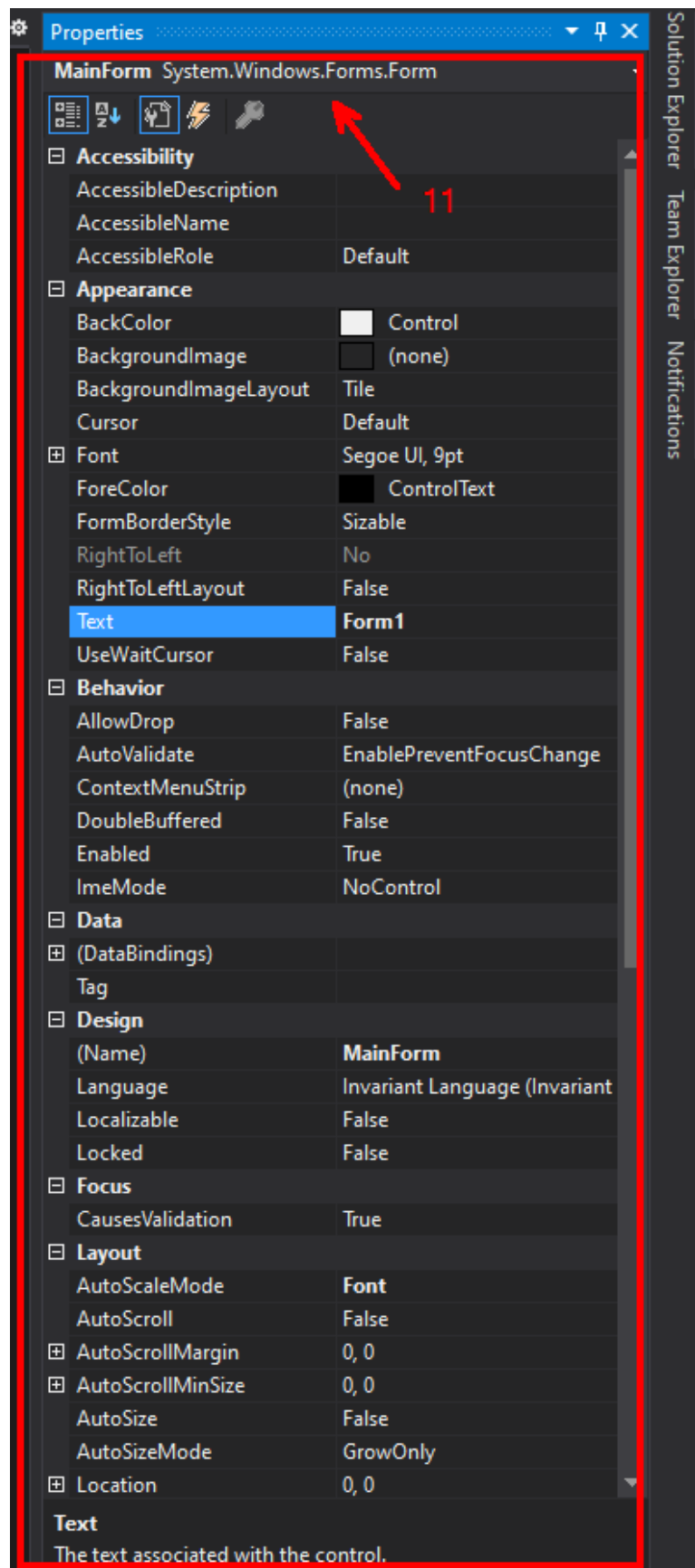
8- Избираме Toolbox



- 9- Менюто toolbox съдържа всичките ни нужни компоненти, за да създадем едно приложение с графичен потребителски интерфейс. Чрез селекция и влачене, те могат да бъдат включвани към Design на формата



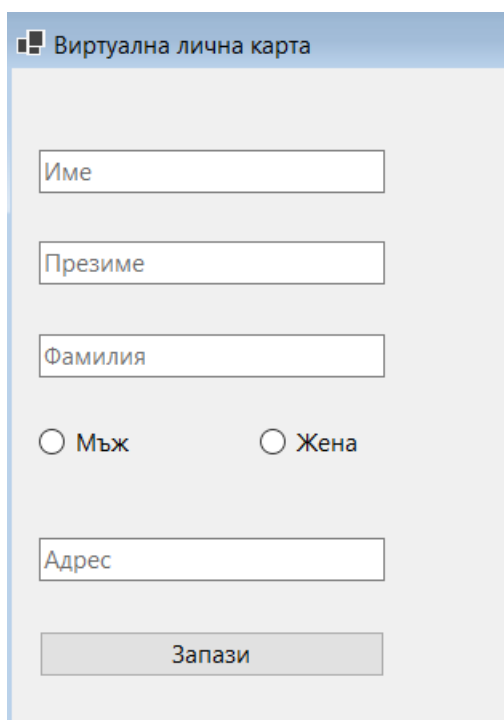
10- Когато сте в режим на Design, натиснете десен бутон на мишката на празно пространство и изберете Properties



11- Менюто Properties, дава възможност да коригирате настройките на селектираният обект

### Задача 21: VirtualID

Направете програма с графичен потребителски интерфейс, която представлява форма за попълване на лична информация. Вашата форма трябва да изглежда по следния начин:



The image shows a graphical user interface for a 'Virtual ID Card' (Виртуална лична карта). The form is contained within a window with a blue title bar. It features five text input fields: 'Име' (Name), 'Презиме' (Surname), 'Фамилия' (Family Name), 'Адрес' (Address), and a 'Запази' (Save) button at the bottom. There are also two radio buttons for gender selection: 'Мъж' (Male) and 'Жена' (Female).

Полетата Име, Презиме, Фамилия и Адрес са текстови, а Мъж и Жена трябва да бъдат радио бутони.

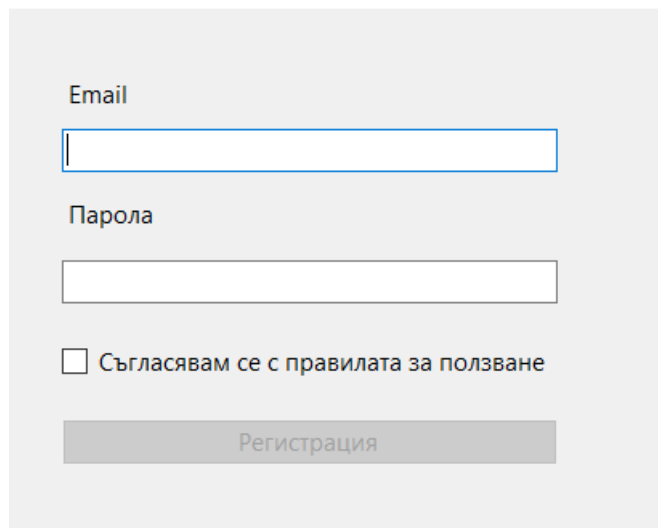
Поведението на вашата програма трябва да бъде следното:

1. Попълване на полетата
  2. Избиране на опция за пол
  3. Натискане на бутона „Запази“
  4. При натискане на ботона да изкочи съобщение, което обобщава въведените данни
- Проучете как и защо използваме класа MessageBox.

### Задача 22: RegisterForm

Направете програма с графичен потребителски интерфейс, която представлява форма за регистрация в система. Вашата форма трябва да изглежда по следния начин:

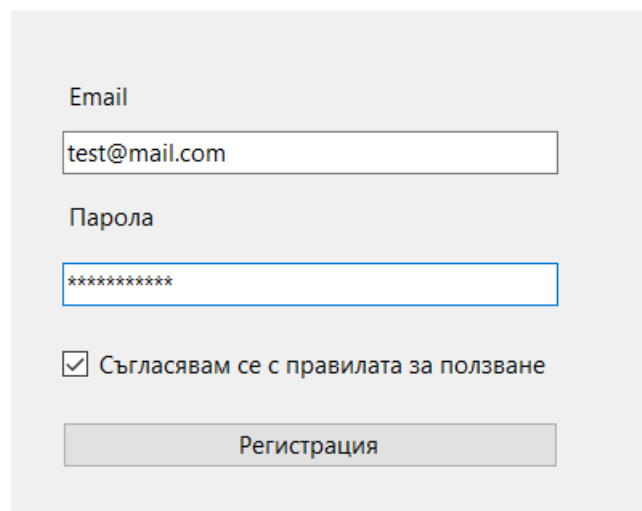
## ■ Регистрация



Registration form with the following elements:

- Email input field (empty)
- Парола (Password) input field (empty)
- ☐ Съгласявам се с правилата за ползване
- Регистрация button (disabled)

## ■ Регистрация



Registration form with the following elements:

- Email input field (filled with test@mail.com)
- Парола (Password) input field (filled with masked characters)
- ☒ Съгласявам се с правилата за ползване
- Регистрация button (disabled)

Полетата Email и Парола са текстови, като полето Парола е защитено със специален символ „\*“.

Поведението на вашата програма трябва да бъде следното:

1. Попълване на полетата

Случай 1:

2. Съгласявате се с общите условия
3. Натискане на бутона „Регистрация“
4. При натискане на бутона да изкочи съобщение, което поздравява потребителя за успешно извършената регистрация



Случай 2:

2. Не се съгласявате с общите условия
3. Бутона „Регистрация“ остава неактивен

### Задача 23: FiguresArea

Направете програма с графичен потребителски интерфейс, която представлява форма за смятане лицето на три различни геометрични фигури (Триъгълник, Правоъгълник и Трапец). Вашата форма трябва да изглежда по следния начин:

Поведението на вашата програма трябва да бъде следното:

1. Избирате вида на геометричната фигура, чието лице искате да бъде изчислено
2. Спрямо селектирането от радио бутона, под формата се визуализират различни полета спрямо нужните параметри за смятане на лицето

Пример – За смятане лицето на триъгълник са ни нужни страна и височина към страната:

Избери вид фигура

☒ Триъгълник      ☐ Правоъгълник      ☐ Трапец

Лице на избраната фигура      161

4. При натискане на бутона „Изчисли“ да се изпише до етикета „Лице на избраната фигура“ числото след извършените калкулации спрямо въведените от потребителя параметри

## IV. Контрол и оценяване на усвоените знания

След като се запознахме с всички основни неща, свързани с програмиране на ниво въведение, е време да проверим усвоените знания до момента.

### Задача 1: CircleArea

Да се създаде програма, която смята лицето на окръжност. Стойността на радиуса да бъде генериран случайно чрез класа Random в диапазона [33, 187]. Формулата да бъде реализирана чрез помощта на класа Math. Резултатът да бъде отпечатан на конзолата и да бъде закръглен до втория знак след десетичната запетая.

Желано поведение на програмата:

```
Microsoft Visual Studio Debug Console
The circle's random height is: 130
The circle's area is: 53092.92

Press any key to close this window . . .
```

## Задача 2: AverageNumber

Да се въведат от клавиатурата произволен брой ненулеви числа в интервала [1, 255] (трябва да бъде направена проверка дали даденото число е в този диапазон), за край на въвеждането служи числото 0. Да се отпечата средно аритметичното на тези от тях, които са четни. Ако нито едно от въведените числа не отговаря на условието, да се отпечата на конзолата съобщение за това.

Желано поведение на програмата:

```
Microsoft Visual Studio Debug Console
Enter some numbers. When you're done, enter 0.
Enter a number:
2
Enter a number:
3
Enter a number:
4
Enter a number:
5
Enter a number:
6
Enter a number:
7
Enter a number:
8
Enter a number:
9
Enter a number:
0
The average of the even numbers between 1 and 255 is: 5

Press any key to close this window . . .
```

```
Microsoft Visual Studio Debug Console
Enter some numbers. When you're done, enter 0.
Enter a number:
3
Enter a number:
43
Enter a number:
43
Enter a number:
45
Enter a number:
55
Enter a number:
0
There aren't any even numbers between 1 and 255.

Press any key to close this window . . .
```

### Задача 3: ExtraordinaryCalculator

Направете програма, която взима от конзолата две числа, които могат да бъдат и цели и с плаваща запетая. След това трябва да взима като вход операцията, която може да бъде (+, -, \*, /):

- Ако операцията е събиране, сумата от двете числа трябва да бъде увеличена с 10%
- Ако операцията е изваждане, намерете по-голямото от двете числа и го повдигнете на квадрат и след това извършете изваждането  $((\text{maxNumber})^2 - (\text{minNumber}))$
- Ако операцията е умножение, умножете първото число по корен квадратен на второто
- Ако операцията е деление, направете проверка дали второто число е 0, понеже не можем да делим на нула извадете съобщение за грешка

Накрая обявения резултат трябва да бъде закръглен до втория знак след десетичната запетая. Вашата програма трябва да извежда съобщение дали иска да бъде прекъсната.

Желано поведение на програмата:

```
Microsoft Visual Studio Debug Console
Do you want to make some calculations? yes/no
dsasdadas
Do you want to make some calculations? yes/no
dsadads
Do you want to make some calculations? yes/no
dsadsad
Do you want to make some calculations? yes/no
no

Press any key to close this window . . .
```

```
C:\Users\Mladenov\Desktop\Github Repos\Thesis\IntroToProgrammingCSharpTasks\ExtraordinaryCalculator\bin\Debug\netcoreapp3.1\ExtraordinaryC...
Do you want to make some calculations? yes/no
dsadads
Do you want to make some calculations? yes/no
yes
Enter first number:
10
Enter second number:
2
Enter an operation (+, -, * or /).
+
The sum increased by 10% is: 13.20
Do you want to make some calculations? yes/no
```

```
C:\Users\Mladenov\Desktop\Github Repos\Thesis\IntroToProgrammingCSharpTasks\ExtraordinaryCalculator\bin\Debug\netcoreapp3.1\ExtraordinaryC...
Do you want to make some calculations? yes/no
yes
Enter first number:
5
Enter second number:
2
Enter an operation (+, -, * or /).
-
(MaxNumber * MaxNumber) - MinNumber is: 23.00
Do you want to make some calculations? yes/no
```

```
C:\Users\Mladenov\Desktop\Github Repos\Thesis\IntroToProgrammingCSharpTasks\ExtraordinaryCalculator\bin\Debug\netcoreapp3.1\ExtraordinaryC...
Do you want to make some calculations? yes/no
yes
Enter first number:
9
Enter second number:
4
Enter an operation (+, -, * or /).
*
First Number * Sqrt(Second Number) is: 18.00
Do you want to make some calculations? yes/no
```

```
C:\Users\Mladenov\Desktop\Github Repos\Thesis\IntroToProgrammingCSharpTasks\ExtraordinaryCalculator\bin\Debug\netcoreapp3.1\ExtraordinaryC...
Do you want to make some calculations? yes/no
yes
Enter first number:
666
Enter second number:
0
Enter an operation (+, -, * or /).
/
Division by 0 is forbidden!
Do you want to make some calculations? yes/no
```

#### Задача 4: JupiterPancakes

Интересен факт: През 1972, Весна Вулович, стюардеса, оцелява след като пада от 10 160 метра.

Вие сте космически инженер, част от екипажа на Мат Деймън (от филма „Марсианецът“ или пък вие сте самия марсианец) за следващата мисия до Юпитер.

Имайки предвид случая на оцелял след падане от почти 10км височина, Мат Ви е задал отговорната задача да проверите дали даден човек е подходящ за вашият екип, като изчислите каква ще бъде неговата енергия на сблъсък, при достигане повърхността на планетата Юпитер.

Ако енергията на сблъсъка надвишава 1 500 000 джаула, човекът не е подходящ за вашия екип, тоест той би се размазал като палачинка на повърхността на планетата – от там и нарицателното Jupiter pancakes (палачинки на Юпитер).

Напишете програма, която чете 1 знаков низ - името на кандидата и 1 цяло число - теглото на кандидата, отделени с един спейс - “ ”.

Създайте методите:

CalcultateHeight(), който изчислява височината на падане, която е равна на сбора от буквите на името на кандидата;

EnergyAtImpact(), който изчислява енергията на сблъсък по следната формула:

$$\text{Energy at impact} = \text{mass} * \text{gravity} * \text{height}$$

**mass** – теглото на човека

**gravity** – гравитацията на Юпитер (използвайте 24.79 m/s<sup>2</sup>)

**height** – височината на падане

Вход	Изход
Petar 70	Energy at impact: 881 532.4 joules. The candidate is approved!
Gosho 82	Energy at impact: 1 040 783.36 joules. The candidate is approved!
Sasho 120	Energy at impact: 1 517 148 joules. The candidate is a “Jupiter pancake”!
Mariq 56	Energy at impact: 702 449.44 joules. The candidate is approved!

Желано поведение на програмата:

```

Microsoft Visual Studio Debug Console
Enter your name and your weight (separate them with a space):
Petar 70
Energy at impact: 881532.4 joules
The candidate is approved!

Press any key to close this window . . .

```



```
Microsoft Visual Studio Debug Console
Enter your name and your weight (separate them with a space):
Sasho 120
Energy at impact: 1517147.9999999998 joules
The candidate is a "Jupiter pancake"!

Press any key to close this window . . .
```

### Задача 5: RainingStations

Три, от най-големите ни станции за засичане на валежите, години на ред записват и смятат получените данни ръчно, но както сами разбирате живеем в 21-ви век и губенето на човешки ресурс за работа, която може да бъде автоматизирана чрез компютър е недопустимо. Затова те се обръщат към вас с молбата да разработите софтуер, който да замести ръчните сметки и изготвянето на дадени статистики.

Напишете програма, която симулира работата на станции засичащи валежите през годината. Във вашата програма трябва да създадете три станции съответно за градовете София, Пловдив и Бургас. Във всяка една от станциите трябва да се пази информация за количеството на валежите за всеки месец през годината. Реализирайте метод, който да:

- Въвежда количество на валежите за всеки месец на случаен принцип в диапазона  $[0, 200]$  за дадена станция
- Да принтира количеството на валежите за всеки месец за дадена станция
- Да намира и отпечатва средното количество на валежите за година за дадена станция
- Да намира и отпечатва месеците, в които има валежи над  $76 \text{ mm/m}^2$  и количеството на валежите се дели на 7 без остатък за дадена станция, а ако няма такива също да се извежда съобщение за това

Дебелината на водния слой (в mm воден стълб), който може да се образува над хоризонтална непропусклива повърхност. Тази единица за количество е идентична на използваната в практиката литър на квадратен метър ( $\text{l/m}^2$ ).

Направете меню за достъпване на всички методи за определена станция, както е дадено в примера долу.

Желано поведение на програмата:

```
Microsoft Visual Studio Debug Console
Choose a station:
1 --> Sofia station
2 --> Plovdiv station
3 --> Burgas station
Enter your option here: 2
-----
Plovdiv Staion
-----
Raining in month 1: 136 mm/m2
Raining in month 2: 130 mm/m2
Raining in month 3: 43 mm/m2
Raining in month 4: 77 mm/m2
Raining in month 5: 188 mm/m2
Raining in month 6: 119 mm/m2
Raining in month 7: 39 mm/m2
Raining in month 8: 196 mm/m2
Raining in month 9: 63 mm/m2
Raining in month 10: 80 mm/m2
Raining in month 11: 41 mm/m2
Raining in month 12: 52 mm/m2
The average amount of rain for a year in this station is: 97.00 mm/m2
Month 4 has amount of rain more than 76 mm/m2 and that amount can be divided by 7 without remainder.

Press any key to close this window . . .
```

## Задача 6: SimpleArray

Прочете от конзолата цели числа и ги запазете в масив. На следващия ред ще получите различни команди (swap {index1} {index 2}, increase {string}, decrease {number} ) за край на програмата служи “End”, след което трябва да се изпечатат елементите на масива.

Команди:

- swap {index1} {index 2} – след командата трябва да получите две числа, да намерете елементите, които се съдържат на тези индекси в масива и размените техните стойности.
- increase {string} – след командата трябва да имате низ, на когото намерете сумата от неговите елементи и след това увеличете всеки един елемент от масива с общата сума от низа.
- decrease {number} – след командата трябва получите число, с което да намалите всеки един от елементите на масива.
- info - след командата трябва да изкарате колко елемента има масива, най-голяма и най-малката стойност също така общата сума на елементите.

За всяка една от командите трябва да имате метод и също така да отпечвате резултата на масива след всяка команда.

Желано поведение на програмата:

```
C:\Users\Mladenov\Desktop\Github Repos\Thesis\IntroToProgrammingCSharpTasks\SimpleArray\bin\Debug\netcoreapp3.1\SimpleArray.exe
Enter number of elements: 6
Number 1: 22
Number 2: 55
Number 3: 3
Number 4: 56
Number 5: 12
Number 6: 31
Choose a command (swap, increase, decrease or info) or type 'End' if you want to exit:
increase
Enter a word:
dell
d --> 100
e --> 101
l --> 108
l --> 108
Total sum: 417
Number 1: 439
Number 2: 472
Number 3: 420
Number 4: 473
Number 5: 429
Number 6: 448
Choose a command (swap, increase, decrease or info) or type 'End' if you want to exit:
-
```

```
Microsoft Visual Studio Debug Console
Number 2: 472
Number 3: 420
Number 4: 473
Number 5: 429
Number 6: 448
Choose a command (swap, increase, decrease or info) or type 'End' if you want to exit:
info
Array count --> 6
Max element --> 473
Min element --> 420
Total sum --> 2681
Number 1: 439
Number 2: 472
Number 3: 420
Number 4: 473
Number 5: 429
Number 6: 448
Choose a command (swap, increase, decrease or info) or type 'End' if you want to exit:
End
Number 1: 439
Number 2: 472
Number 3: 420
Number 4: 473
Number 5: 429
Number 6: 448
Press any key to close this window . . .
```

## V. Заключение

Тази разработка показва един методически подход, чрез който може да се преподава учебно съдържание по информатика за общообразователна подготовка. Дипломната работа включва теоретично съдържание по програмиране, разнообразни примери, разнородни учебни задачи за упражнение, задачи за контрол и оценяване на знанията на учениците. Информацията в дипломната работа отговаря напълно на държавните образователни стандарти и учебни програми по информатика за общообразователна подготовка.

Всички решения на задачите са реализирани с помощта на интегрираната среда за разработка на софтуер „Visual Studio 2019“ и езика за програмиране „C# .NET Core 3.1“. Задачите представени в тази разработка са по-скоро концептуални, отколкото обвързани изцяло с технологията на C#, защото предлагат предизвикателни условия, които имат за цел прилагането на всички знания и умения на учениците, усвоени до момента.

Системата от задачи по програмиране, описана в дипломната работа, се състои от 23 учебни задачи и 6 задачи за контрол и оценяване на знанията на учениците. Считаме, че избраните задачи реализират идеите заложи в учебната програма по информатика в VIII клас на СУ.

Перспективите за развитие на дипломната работа са да се разработват подобни учебни помагала, посветени и на други езици за програмиране като Java, C++, Python и други.

Всички задачи от дипломната работа и техните решения могат да се намерят на интернет адрес [12]: <https://github.com/deivid7007/Thesis>

## VI. Използвана литература

1. Ангелов, А., К. Гъров, О. Гавраилов, Информатика за 10 клас на ЕСПУ, изд. Народна просвета, София, 1986.
2. Ангелов, А., К. Гъров, О. Гавраилов, Информатика за 11 клас на ЕСПУ, изд. Народна просвета, София, 1987.
3. Бърнев П., П. Азълов, Д. Добрев, Ц. Бистеров, Информатика за 10 клас на ЕСПУ, изд. Народна просвета, София, 1986.
4. Бърнев П., П. Азълов, Д. Добрев, Ц. Бистеров, Информатика за 11 клас на ЕСПУ, изд. Народна просвета, София, 1987.
5. Гавраилов О., К. Гъров, Информатика, изд. “АСИО”, София, 1994.
6. Държавни образователни изисквания за учебно съдържание в СОУ, Държавен вестник, бр. 43, 18. 06. 2000.
7. К. Гъров, Някои методически аспекти на обучението по информатика и информационни технологии, Пловдив 2013.
8. Новите учебници по информатика и информационни технологии, сп. Математика и информатика, бр. 5 и 6, София, 2001.
9. Указание за организиране на обучението по ИТ в средните общообразователни училища и гимназиите през учебната 1994/95 година, изд. на МНО, София, 1994.
10. Учебни програми по информатика и информационни технологии, сп. Математика и информатика, бр. 3-4, стр.24 – 90, София, 2001.
11. <https://github.com/BaiGanio/IntroCSharp> (посетен на 06.07.2020 г.)
12. <https://github.com/deivid7007/Thesis> (посетен на 06.07.2020 г.)
13. <https://introprogramming.info/intro-csharp-book> (посетен на 06.07.2020 г.)
14. [https://nssdc.gsfc.nasa.gov/planetary/factsheet/planet\\_table\\_ratio.html](https://nssdc.gsfc.nasa.gov/planetary/factsheet/planet_table_ratio.html) (посетен на 06.07.2020 г.)
15. <https://www.wikipedia.org/> (посетен на 06.07.2020 г.)

## VII. Приложение с решения на задачите от дипломната работа

### 1. Решения на задачите от разработените теми

#### Задача 1:

Решението на тази задача присъства в демото към тема „Създаване на решение (Solution)“.

#### Задача 2: TheHiProgram

```
using System;

namespace TheHiProgram
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello! I am Deivid Mladenov");
        }
    }
}
```

#### Задача 3: NumbersInDifferentNumeralSystems

```
using System;

namespace NumbersInDifferentNumeralSystems
{
    class Program
    {
        static void Main(string[] args)
        {
            int number = 555;

            string binary = Convert.ToString(number, 2);
            string ocataldecimal = Convert.ToString(number, 8);
            string hexadecimal = Convert.ToString(number, 16);

            Console.WriteLine($"The number 555 in binary is {binary}");
            Console.WriteLine($"The number 555 in ocataldecimal is {ocataldecimal}");
            Console.WriteLine($"The number 555 in hexadecimal is {hexadecimal}");
        }
    }
}
```

#### Задача 4: AgeAfter10Years

```
using System;

namespace AgeAfter10Years
{
    class Program
    {
        static void Main(string[] args)
        {
            int age = 0;
            Console.WriteLine("How old are you?");
            age = int.Parse(Console.ReadLine());
            Console.WriteLine($"You will be {age+10} years old after 10 years!");
        }
    }
}
```

#### Задача 5: TheUniqueNumber

```
using System;

namespace TheUniqueNumber
{
    class Program
    {
        static void Main(string[] args)
        {
            int ageAfter10Years = 0;
            Console.WriteLine("How old are you?");
            ageAfter10Years = int.Parse(Console.ReadLine()) + 10;
            Console.WriteLine("Which is your birth year?");
            int birthYear = int.Parse(Console.ReadLine());
            Console.WriteLine("Which is your birth month?");
            int birthMonth = int.Parse(Console.ReadLine());
            double uniqueNumber = (ageAfter10Years * birthYear) / Math.PI *
            birthMonth;
            Console.WriteLine(
                $"Your unique number is {Math.Round(uniqueNumber, 4)}!");
        }
    }
}
```

### Задача 6: TravelingToTheSunWithSoundSpeed

```
using System;

namespace TravelingToTheSunWithSoundSpeed
{
    class Program
    {
        static void Main(string[] args)
        {
            const double speedOfSound = 1225.044; // in km/h
            const long distToSun = 149600000; // in km
            double timeNeeded = Math.Round(distToSun / speedOfSound); // in hours
            double timeNeededInYears = Math.Floor(timeNeeded / 8760);
            double leftoverDays = Math.Floor(((timeNeeded / 8760) -
            timeNeededInYears) * 365);

            Console.WriteLine($"The time needed to travel from the Earth to the Sun is
            {timeNeeded} hours, or "
            + $"{timeNeededInYears} years and {leftoverDays} days.");
        }
    }
}
```

### Задача 7: OddOrEven

```
using System;

namespace OddOrEven
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter your number?");
            int number = int.Parse(Console.ReadLine());

            if (number % 2 == 0)
            {
                Console.WriteLine("The number is even!");
            }
            else
            {
                Console.WriteLine("The number is odd!");
            }
        }
    }
}
```



### Задача 8: NumberComparison

```
using System;

namespace NumberComparison
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter first number:");
            int firstNumber = int.Parse(Console.ReadLine());
            Console.WriteLine("Enter second number:");
            int secondNumber = int.Parse(Console.ReadLine());

            if (firstNumber > secondNumber)
            {
                Console.WriteLine($"{firstNumber} is greater than {secondNumber}!");
            }
            else if (firstNumber == secondNumber)
            {
                Console.WriteLine($"{firstNumber} is equal to {secondNumber}!");
            }
            else
            {
                Console.WriteLine($"{firstNumber} is less than {secondNumber}!");
            }
        }
    }
}
```

### Задача 9: MyNumberDiapason

```
using System;

namespace MyNumberDiapason
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter your number:");
            int number = int.Parse(Console.ReadLine());

            if (number < 1000)
            {
                Console.WriteLine($"The number {number} is less than 1000!");
            }
            else if (number > 1000 && number < 10000)
            {
                Console.WriteLine($"The number {number} is between 1000 and 10000!");
            }
            else
            {
                Console.WriteLine($"The number {number} is greater than 10000!");
            }
        }
    }
}
```

### Задача 10: CalculateMyWeightInSolarSystem

```
using System;

namespace CalculateMyWeightInSolarSystem
{
    class Program
    {
        static void Main(string[] args)
        {
            double gravitationalRatio;
            double weightOnPlanet;
            Console.WriteLine("-----");
            Console.WriteLine("Enter the code corresponding to the name of the desired
planet.");
            Console.WriteLine("1: Mercury");
            Console.WriteLine("2: Venus");
            Console.WriteLine("3: Earth (You already know the answer)");
            Console.WriteLine("4: Mars");
            Console.WriteLine("5: Jupiter");
            Console.WriteLine("6: Saturn");
            Console.WriteLine("7: Uranus");
            Console.WriteLine("8: Neptune");
            Console.WriteLine("9: Pluto");
            Console.WriteLine("-----");

            byte option = byte.Parse(Console.ReadLine());

            Console.WriteLine("Enter your weight in kgs as measured on Earth.");
            byte weight = byte.Parse(Console.ReadLine());

            switch (option)
            {
                case 1:
                    gravitationalRatio = 0.38;
                    weightOnPlanet = gravitationalRatio * weight;
                    Console.WriteLine($"You weigh approximately {weightOnPlanet}kg on
Mercury.");
                    break;
                case 2:
                    gravitationalRatio = 0.91;
                    weightOnPlanet = gravitationalRatio * weight;
                    Console.WriteLine($"You weigh approximately {weightOnPlanet}kg on
Venus.");
                    break;
                case 3:
                    gravitationalRatio = 1;
                    weightOnPlanet = gravitationalRatio * weight;
```

```

        Console.WriteLine($"You weigh approximately {weightOnPlanet}kg on
Earth.");
        break;
    case 4:
        gravitationalRatio = 0.38;
        weightOnPlanet = gravitationalRatio * weight;
        Console.WriteLine($"You weigh approximately {weightOnPlanet}kg on
Mars.");
        break;
    case 5:
        gravitationalRatio = 2.36;
        weightOnPlanet = gravitationalRatio * weight;
        Console.WriteLine($"You weigh approximately {weightOnPlanet}kg on
Jupiter.");
        break;
    case 6:
        gravitationalRatio = 0.92;
        weightOnPlanet = gravitationalRatio * weight;
        Console.WriteLine($"You weigh approximately {weightOnPlanet}kg on
Saturn.");
        break;
    case 7:
        gravitationalRatio = 0.89;
        weightOnPlanet = gravitationalRatio * weight;
        Console.WriteLine($"You weigh approximately {weightOnPlanet}kg on
Uranus.");
        break;
    case 8:
        gravitationalRatio = 1.12;
        weightOnPlanet = gravitationalRatio * weight;
        Console.WriteLine($"You weigh approximately {weightOnPlanet}kg on
Neptune.");
        break;
    case 9:
        gravitationalRatio = 0.07;
        weightOnPlanet = gravitationalRatio * weight;
        Console.WriteLine($"You weigh approximately {weightOnPlanet}kg on
Pluto.");
        break;
    default: Console.WriteLine("Enter a valid code."); break;
    }
}
}
}
}

```

### Задача 11: SimpleDeckOfCards

using System;

namespace SimpleDeckOfCards

{

class Program

{

static void Main(string[] args)

{

Console.WriteLine("Do you want to see a full deck of cards? (yes or no)");

string answer = Console.ReadLine();

while (answer != "yes")

{

Console.WriteLine("Do you want to see a full deck of cards? (yes or no)");

answer = Console.ReadLine();

}

Console.WriteLine(\$"A {\u2660}, A {\u2665}, A {\u2666}, A  
{\u2663}");

Console.WriteLine(\$"2 {\u2660}, 2 {\u2665}, 2 {\u2666}, 2 {\u2663}");

Console.WriteLine(\$"3 {\u2660}, 3 {\u2665}, 3 {\u2666}, 3 {\u2663}");

Console.WriteLine(\$"4 {\u2660}, 4 {\u2665}, 4 {\u2666}, 4 {\u2663}");

Console.WriteLine(\$"5 {\u2660}, 5 {\u2665}, 5 {\u2666}, 5 {\u2663}");

Console.WriteLine(\$"6 {\u2660}, 6 {\u2665}, 6 {\u2666}, 6 {\u2663}");

Console.WriteLine(\$"7 {\u2660}, 7 {\u2665}, 7 {\u2666}, 7 {\u2663}");

Console.WriteLine(\$"8 {\u2660}, 8 {\u2665}, 8 {\u2666}, 8 {\u2663}");

Console.WriteLine(\$"9 {\u2660}, 9 {\u2665}, 9 {\u2666}, 9 {\u2663}");

Console.WriteLine(\$"10 {\u2660}, 10 {\u2665}, 10 {\u2666}, 10  
{\u2663}");

Console.WriteLine(\$"J {\u2660}, J {\u2665}, J {\u2666}, J {\u2663}");

Console.WriteLine(\$"Q {\u2660}, Q {\u2665}, Q {\u2666}, Q  
{\u2663}");

Console.WriteLine(\$"K {\u2660}, K {\u2665}, K {\u2666}, K  
{\u2663}");

}

}

}

## Задача 12: BusTicketLuckyNumber

```
using System;

namespace BusTicketLuckyNumber
{
    class Program
    {
        static void Main(string[] args)
        {
            int luckyTicketsCount = 0;
            int unluckyTicketsCount = 0;

            for (int i = 0; i < 1000000; i++)
            {
                if ((i / 100000 + i % 100000 / 10000 + i % 100000 % 10000 / 1000) == (i % 100000 % 10000 % 1000 / 100 + i % 100000 % 10000 % 1000 % 100 / 10 + i % 10))
                {
                    Console.WriteLine(i);
                    luckyTicketsCount++;
                }
                else
                {
                    unluckyTicketsCount++;
                }
            }

            Console.WriteLine("The number of lucky tickets is: " + luckyTicketsCount);
            Console.WriteLine("The number of unlucky tickets is: " +
unluckyTicketsCount);
            Console.WriteLine($"Total tickets: {luckyTicketsCount +
unluckyTicketsCount}");
        }
    }
}
```

### Задача 13: MiserPiggy

```
using System;

namespace MiserPiggy
{
    class Program
    {
        static void Main(string[] args)
        {
            int sumPiggyName1 = 0;
            int sumPiggyName2 = 0;
            int sumPiggyName3 = 0;

            Console.WriteLine("Hey, piggies! Enter your names:");
            Console.Write("Piggy 1: ");
            string piggyName1 = Console.ReadLine();
            Console.Write("Piggy 2: ");
            string piggyName2 = Console.ReadLine();
            Console.Write("Piggy 3: ");
            string piggyName3 = Console.ReadLine();

            foreach (char a in piggyName1)
            {
                sumPiggyName1 += a;
            }

            foreach (char b in piggyName2)
            {
                sumPiggyName2 += b;
            }

            foreach (char c in piggyName3)
            {
                sumPiggyName3 += c;
            }

            if (sumPiggyName1 > sumPiggyName2 && sumPiggyName1 >
sumPiggyName3)
            {
                if (sumPiggyName2 > sumPiggyName3)
                {
                    Console.WriteLine($"The name-wealthiest pig is {piggyName1 }
({sumPiggyName1 }) followed by " +
                    $"{piggyName2} ({sumPiggyName2}) and last is {piggyName3}
({sumPiggyName3}).");
                }
                else
                {

```

```

        Console.WriteLine($"The name-wealthiest pig is {piggyName1}
({sumPiggyName1}) followed by " +
        $"{piggyName3} ({sumPiggyName3}) and last is {piggyName2}
({sumPiggyName2}).");
    }
}
else if (sumPiggyName2 > sumPiggyName1 && sumPiggyName2 >
sumPiggyName3)
{
    if (sumPiggyName1 > sumPiggyName3)
    {
        Console.WriteLine($"The name-wealthiest pig is {piggyName2}
({sumPiggyName2}) followed by " +
        $"{piggyName1} ({sumPiggyName1}) and last is {piggyName3}
({sumPiggyName3}).");
    }
    else
    {
        Console.WriteLine($"The name-wealthiest pig is {piggyName2}
({sumPiggyName2}) followed by " +
        $"{piggyName3} ({sumPiggyName3}) and last is {piggyName1}
({sumPiggyName1}).");
    }
}
else if (sumPiggyName1 > sumPiggyName2)
{
    Console.WriteLine($"The name-wealthiest pig is {piggyName3}
({sumPiggyName3}) followed by " +
    $"{piggyName1} ({sumPiggyName1}) and last is {piggyName2}
({sumPiggyName2}).");
}
else
{
    Console.WriteLine($"The name-wealthiest pig is {piggyName3}
({sumPiggyName3}) followed by " +
    $"{piggyName2} ({sumPiggyName2}) and last is {piggyName1}
({sumPiggyName1}).");
}
}
}
}

```



#### Задача 14: WorkingWithSomeIntegers

```
using System;

namespace WorkingWithSomeIntegers
{
    class Program
    {
        static Random rand = new Random();
        static void Main(string[] args)
        {
            int[] intArray = new int[100];
            Console.WriteLine("Here are 100 randomly generated numbers in the interval  
between 0 and 132.");
            for (int i = 0; i < intArray.Length; i++)
            {
                intArray[i] = rand.Next(132);
                Console.WriteLine(intArray[i]);
            }
            Console.WriteLine("Choose what you want to do with them (type in the  
letter): ");
            Console.WriteLine("a) Display the numbers that have even indexes.");
            Console.WriteLine("b) Display the numbers that are odd and have odd  
indexes.");
            Console.WriteLine("c) Display the numbers that are divided by 3 without a  
remainder.");
            Console.WriteLine("d) Display the numbers that are divided by 7 and have 1  
as remainder.");
            Console.WriteLine("e) Display the numbers that are in the interval between 26  
and 100.");
            Console.WriteLine("f) Display the numbers that are not in the interval  
between 26 and 100.");

            Console.WriteLine("-----");
            char option = char.Parse(Console.ReadLine());
            Console.WriteLine("-----");

            switch (option)
            {
                case 'a':
                    for (int i = 0; i < intArray.Length; i++)
                    {
                        if (i % 2 == 0)
                        {
                            Console.WriteLine($"{intArray[i]} has index {i}.");
                        }
                    }
                    break;
                case 'b':
                    for (int i = 0; i < intArray.Length; i++)
```

```

        {
            if ((intArray[i] % 2) != 0 && (i % 2 != 0))
                Console.WriteLine($"{intArray[i]} is odd and has index {i}.");
        }
        break;
    case 'c':
        for (int i = 0; i < intArray.Length; i++)
        {
            if (intArray[i] % 3 == 0)
                Console.WriteLine($"{intArray[i]} is divided by 3 without a
remainder.");
        }
        break;
    case 'd':
        for (int i = 0; i < intArray.Length; i++)
        {
            if (intArray[i] % 7 == 1)
                Console.WriteLine($"{intArray[i]} is divided by 7 with a remainder
of 1.");
        }
        break;
    case 'e':
        for (int i = 0; i < intArray.Length; i++)
        {
            if ((intArray[i] >= 26) && (intArray[i] <= 100))
                Console.WriteLine($"{intArray[i]} is between 26 and 100.");
        }
        break;
    case 'f':
        for (int i = 0; i < intArray.Length; i++)
        {
            if ((intArray[i] < 26) || (intArray[i] > 100))
                Console.WriteLine($"{intArray[i]} is not between 26 and 100.");
        }
        break;
    default:
        Console.WriteLine("Incorrect option!");
        break;
    }
}
}
}

```

### Задача 15: WorkingWithSomeLetters

```
using System;
using System.Text;

namespace WorkingWithSomeLetters
{
    class Program
    {
        static Random rand = new Random();

        static void Main(string[] args)
        {
            Console.OutputEncoding = Encoding.UTF8;

            string[] stringArray = new string[100];

            Console.WriteLine("Даден е низ от 100 случайни букви на кирилица.  
Програмата показва кои гласни са широки, тесни и съставни, както и кои  
съгласни са звучни, беззвучни и сонорни.");
            Console.WriteLine("-----");

            for (int i = 0; i < 100; i++)
            {
                stringArray[i] = ((char)rand.Next(1072, 1103)).ToString();
                Console.Write(stringArray[i]);
            }

            Console.WriteLine("\n-----");

            for (int i = 0; i < 100; i++)
            {
                if ((stringArray[i] == "а") || (stringArray[i] == "о") || (stringArray[i] ==  
"е"))
                {
                    Console.WriteLine($"Буквата ({stringArray[i]}) е широка гласна и се  
намира под номер {i}.");
                }
                if ((stringArray[i] == "и") || (stringArray[i] == "у") || (stringArray[i] ==  
"ъ"))
                {
                    Console.WriteLine($"Буквата ({stringArray[i]}) е тясна гласна и се  
намира под номер {i}.");
                }
                if ((stringArray[i] == "ю") || (stringArray[i] == "я"))
                {
                    Console.WriteLine($"Буквата ({stringArray[i]}) е съставна гласна и се  
намира под номер {i}.");
                }
            }
        }
    }
}
```

```

        if ((stringArray[i] == "п") || (stringArray[i] == "ф") || (stringArray[i] ==
"к") || (stringArray[i] == "т") || (stringArray[i] == "ш") || (stringArray[i] == "с") ||
(stringArray[i] == "х") || (stringArray[i] == "ц"))
        {
            Console.WriteLine($"Буквата ({stringArray[i]}) е беззвучна съгласна и
се намира под номер {i}.");
        }
        if ((stringArray[i] == "л") || (stringArray[i] == "м") || (stringArray[i] ==
"н") || (stringArray[i] == "р"))
        {
            Console.WriteLine($"Буквата ({stringArray[i]}) е сонорна съгласна и
се намира под номер {i}.");
        }
        if ((stringArray[i] == "б") || (stringArray[i] == "в") || (stringArray[i] == "г")
|| (stringArray[i] == "д") || (stringArray[i] == "ж") || (stringArray[i] == "з") ||
(stringArray[i] == "ч"))
        {
            Console.WriteLine($"Буквата ({stringArray[i]}) е звучна съгласна и се
намира под номер {i}.");
        }
    }
}
}
}

```

### Задача 16: GetMax

```
using System;

namespace GetMax
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter 3 numbers.");
            Console.Write("Enter the first number: ");
            int firstNumber = int.Parse(Console.ReadLine());
            Console.Write("Enter the second number: ");
            int secondNumber = int.Parse(Console.ReadLine());
            Console.Write("Enter the third number: ");
            int thirdNumber = int.Parse(Console.ReadLine());

            Console.WriteLine("The biggest number is: " + GetMax(firstNumber,
secondNumber, thirdNumber));
        }

        static int GetMax(int num1, int num2, int num3)
        {
            if (num1 > num2 && num1 > num3)
            {
                return num1;
            }
            else if (num2 > num1 && num2 > num3)
            {
                return num2;
            }
            else return num3;
        }
    }
}
```

### Задача 17: GeometryCalculator

```
using System;

namespace GeometryCalculator
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Choose and option (triangle, square or radians:");
            string option = Console.ReadLine();

            switch (option)
            {
                case "triangle":
                    Console.Write("Enter a side: ");
                    double triangleSide = double.Parse(Console.ReadLine());
                    Console.Write("Enter the height to that side: ");
                    double triangleHeight = double.Parse(Console.ReadLine());

                    Console.WriteLine("The area of this triangle is: " +
TriangleArea(triangleSide, triangleHeight));
                    break;
                case "square":
                    Console.Write("Enter a side: ");
                    double squareSide = double.Parse(Console.ReadLine());

                    Console.WriteLine("The area of this square is: " +
SquareArea(squareSide));
                    break;
                case "radians":
                    Console.Write("Enter an angle in degrees: ");
                    double angleDegrees = double.Parse(Console.ReadLine());

                    Console.WriteLine($"{angleDegrees} degrees in radians is: " +
AngleInRadians(angleDegrees));
                    break;
                default:
                    Console.WriteLine("Incorrect option.");
                    break;
            }
        }

        static double TriangleArea(double tSide, double tHeight)
        {
            double triangleArea = (tSide * tHeight) / 2;

            return triangleArea;
        }
    }
}
```

```

static double SquareArea(double aSquare)
{
    double sSquare = aSquare * aSquare;

    return sSquare;
}

static double AngleInRadians(double angleDeg)
{
    double angleInRadians = Math.PI / 180 * angleDeg;

    return angleInRadians;
}
}
}

```

### Задача 18: StringManipulations

```

using System;

namespace StringManipulations
{
    class Program
    {
        static void Main(string[] args)
        {
            string text = "Hello, I am Deivid Mladenov and I appreciate your attention!";
            Console.WriteLine($"Starting text ---> {text}");
            Console.WriteLine($"To lower ---> {text.ToLower()}");
            Console.WriteLine($"To upper ---> {text.ToUpper()}");
            Console.WriteLine($"IndexOf Deivid ---> {text.IndexOf("Deivid")}");
            Console.WriteLine($"Replace attention with work --->
{text.Replace("attention", "work")}");
        }
    }
}

```

### Задача 19: JustCalculator

```
using System;
namespace JustCalculator
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                Console.WriteLine("Please select action (+, -, *, /)");
                char action = char.Parse(Console.ReadLine());
                Console.WriteLine("Enter first number: ");
                int firstNumber = int.Parse(Console.ReadLine());
                Console.WriteLine("Enter second number: ");
                int secondNumber = int.Parse(Console.ReadLine());
                switch (action)
                {
                    case '+':
                        Console.WriteLine($"{firstNumber} + {secondNumber} = {firstNumber + secondNumber}");
                        break;
                    case '-':
                        Console.WriteLine($"{firstNumber} - {secondNumber} = {firstNumber - secondNumber}");
                        break;
                    case '*':
                        Console.WriteLine($"{firstNumber} * {secondNumber} = {firstNumber * secondNumber}");
                        break;
                    case '/':
                        Console.WriteLine($"{firstNumber} / {secondNumber} = {firstNumber / secondNumber}");
                        break;
                    default:
                        Console.WriteLine("Incorrect action!");
                        break;
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"The program threw exception with message: {ex.Message}");
            }
        }
    }
}
```



## Задача 20: UserWithActions

### Program.cs

```
using System;
namespace UserWithActions
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Enter email:");
            string email = Console.ReadLine();
            Console.WriteLine("Enter first name:");
            string firstName = Console.ReadLine();
            Console.WriteLine("Enter last name:");
            string lastName = Console.ReadLine();
            Console.WriteLine("Enter age:");
            int age = int.Parse(Console.ReadLine());
            Console.WriteLine("Enter city:");
            string city = Console.ReadLine();

            User user = new User(email, firstName, lastName, age, city);

            Console.WriteLine("Select option:");
            Console.WriteLine("1) Introduce user");
            Console.WriteLine("2) Is user adult or not");
            Console.WriteLine("3) Where is user from");
            int option = int.Parse(Console.ReadLine());

            switch (option)
            {
                case 1:
                    user.IntroduceMyself();
                    break;
                case 2:
                    user.AmIAadult();
                    break;
                case 3:
                    user.WhereAmIFrom();
                    break;
                default:
                    Console.WriteLine("Incorrect option picked!");
                    break;
            }
        }
    }
}
```

## User.cs

```
using System;

namespace UserWithActions
{
    public class User
    {
        public User(string email, string firstName, string lastName, int age, string city)
        {
            this.Email = email;
            this.FirstName = firstName;
            this.LastName = lastName;
            this.Age = age;
            this.City = city;
        }

        public string Email { get; set; }
        public string FirstName { get; set; }
        public string LastName { get; set; }
        public int Age { get; set; }
        public string City { get; set; }

        public void IntroduceMyself()
        {
            Console.WriteLine($"I am {FirstName + ' ' + LastName}, {Age} years old
and my email is {Email}");
        }

        public void AmIAAdult()
        {
            if (Age >= 18)
            {
                Console.WriteLine("Yes I am adult, because I am equal or above age of
18");
            }
            else
            {
                Console.WriteLine("No I am not adult, because I am under age of 18");
            }
        }

        public void WhereAmIFrom()
        {
            Console.WriteLine($"I am from {City}");
        }
    }
}
```

### Задача 21, 22 и 23:

Решения към тези задачи не са добавени.

## 2. Решения на задачите от контрол и оценяване

### Задача 1: CircleArea

```
using System;

namespace CircleArea
{
    class Program
    {
        static Random rand = new Random();

        static void Main(string[] args)
        {
            const double pi = Math.PI;
            double circleRadius = rand.Next(33, 188);
            double circleArea = pi * circleRadius * circleRadius;

            Console.WriteLine("The circle's random height is: " + circleRadius);
            Console.WriteLine("The circle's area is: {0:F2}", circleArea);
        }
    }
}
```

## Задача 2: AverageNumber

```
using System;

namespace AverageNumber
{
    class Program
    {
        static void Main(string[] args)
        {
            int number;
            int evenNumbersCount = 0;
            double evenNumbersSum = 0;
            bool inputIsNumber;

            Console.WriteLine("Enter some numbers. When you're done, enter 0.");

            do
            {
                Console.WriteLine("Enter a number: ");
                inputIsNumber = int.TryParse(Console.ReadLine(), out number);

                if ((number >= 1 && number <= 255) && (number % 2 == 0))
                {
                    evenNumbersSum += number;
                    evenNumbersCount++;
                }
            } while (number != 0 || !inputIsNumber);

            if (evenNumbersCount == 0)
            {
                Console.WriteLine("There aren't any even numbers between 1 and 255.");
            }
            else
            {
                double evenNumbersAverage = evenNumbersSum / evenNumbersCount;
                Console.WriteLine("The average of the even numbers between 1 and 255
is: " + evenNumbersAverage);
            }
        }
    }
}
```

### Задача 3: ExtraordinaryCalculator

```
using System;
namespace ExtraordinaryCalculator
{
    class Program
    {
        static void Main(string[] args)
        {
            string answer;

            do
            {
                Console.WriteLine("Do you want to make some calculations? yes/no");
                answer = Console.ReadLine();

                if (answer == "yes")
                {
                    double firstNumber = 0;
                    double secondNumber = 0;

                    bool firstInputIsNumber = false;
                    bool secondInputIsNumber = false;

                    while (!firstInputIsNumber)
                    {
                        Console.WriteLine("Enter first number:");
                        firstInputIsNumber = double.TryParse(Console.ReadLine(), out
firstNumber);

                        if (!firstInputIsNumber)
                        {
                            Console.WriteLine("Enter a valid number!");
                        }
                    }

                    while (!secondInputIsNumber)
                    {
                        Console.WriteLine("Enter second number:");
                        secondInputIsNumber = double.TryParse(Console.ReadLine(), out
secondNumber);

                        if (!secondInputIsNumber)
                        {
                            Console.WriteLine("Enter a valid number!");
                        }
                    }

                    Console.WriteLine("Enter an operation (+, -, * or /).");
                    char calcOperator = char.Parse(Console.ReadLine());
```

```

double result;

switch (calcOperator)
{
    case '+':
        double sum = firstNumber + secondNumber;
        result = sum + sum * 0.1;
        Console.WriteLine("The sum increased by 10% is: {0:F2}", result);
        break;
    case '-':
        if (firstNumber > secondNumber)
        {
            result = firstNumber * firstNumber - secondNumber;
        }
        else
        {
            result = secondNumber * secondNumber - firstNumber;
        }
        Console.WriteLine("(MaxNumber * MaxNumber) - MinNumber is:
{0:F2}", result);
        break;
    case '*':
        result = firstNumber * Math.Sqrt(secondNumber);
        Console.WriteLine("First Number * Sqrt(Second Number) is:
{0:F2}", result);
        break;
    case '/':
        if (secondNumber == 0)
        {
            Console.WriteLine("Division by 0 is forbidden!");
        }
        break;
    default:
        Console.WriteLine("Incorrect operation!");
        break;
}
}
} while (answer != "no");

Environment.Exit(0);
}
}
}

```

#### Задача 4: JupiterPancakes

```
using System;
using System.Text;

namespace JupiterPancakes
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.OutputEncoding = Encoding.UTF8;

            bool inputIsCorrect = false;
            bool weightIsCorrect = false;

            string userInput;
            string astronautName = "";

            int weight = 0;

            while (!inputIsCorrect || !weightIsCorrect)
            {
                Console.WriteLine("Enter your name and your weight (separate them with  
a space):");
                userInput = Console.ReadLine();

                if (!userInput.Contains(" "))
                {
                    Console.WriteLine("Incorrect data!");
                    continue;
                }
                else
                {
                    inputIsCorrect = true;

                    astronautName = userInput.Substring(0, userInput.IndexOf(" "));
                    weightIsCorrect = int.TryParse(userInput.Substring(userInput.IndexOf(" ")), out weight);
                }

                int fallHeight = CalculateHeight(astronautName);
                double energy = EnergyAtImpact(weight, fallHeight);

                Console.WriteLine("Energy at impact: " + energy + " joules");

                if (energy < 1500000)
                {
                    Console.WriteLine("The candidate is approved!");
                }
            }
        }
    }
}
```

```

    }
    else
    {
        Console.WriteLine("The candidate is a \"Jupiter pancake\"!");
    }
}

static int CalculateHeight(string name)
{
    int charSum = 0;

    foreach (char c in name)
    {
        charSum += c;
    }

    return charSum;
}

static double EnergyAtImpact(int mass, int height)
{
    const double jupiterGravity = 24.79;

    return mass * jupiterGravity * height;
}
}
}

```



### Задача 5: RainingStations

```
using System;

namespace RainingStations
{
    class Program
    {
        static Random rand = new Random();

        static void Main(string[] args)
        {
            Console.WriteLine("Choose a station:\n" +
                "1 --> Sofia station\n" +
                "2 --> Plovdiv station\n" +
                "3 --> Burgas station");

            bool optionIsNumber = false;
            int option = 0;

            while (!optionIsNumber)
            {
                Console.Write("Enter your option here: ");
                optionIsNumber = int.TryParse(Console.ReadLine(), out option);
            }

            switch (option)
            {
                case 1:
                    Console.WriteLine("-----\n" +
                        "Sofia Staion\n" +
                        "-----");

                    int[] sofiaArray = new int[12];

                    InputArray(sofiaArray);
                    OutputArray(sofiaArray);

                    double averageRainInSofia =
                        AverageAmountOfRainingForAYear(sofiaArray);
                    Console.WriteLine("The average amount of rain for a year in this station
is: {0:F2} mm/m2",
                        averageRainInSofia);

                    PrintMonthsWithFilter(sofiaArray);

                    break;
                case 2:
                    Console.WriteLine("-----\n" +
                        "Plovdiv Staion\n" +
```

```

        "-----");

        int[] plovdivArray = new int[12];
        InputArray(plovdivArray);
        OutputArray(plovdivArray);
        double averageRainInPlovdiv =
AverageAmountOfRainingForAYear(plovdivArray);
        Console.WriteLine("The average amount of rain for a year in this station
is: {0:F2} mm/m2",
            averageRainInPlovdiv);
        PrintMonthsWithFilter(plovdivArray);

        break;
    case 3:
        Console.WriteLine("-----\n" +
            "Burgas Staion\n" +
            "-----");

        int[] burgasArray = new int[12];
        InputArray(burgasArray);
        OutputArray(burgasArray);
        double averageRainInBurgas =
AverageAmountOfRainingForAYear(burgasArray);
        Console.WriteLine("The average amount of rain for a year in this station
is: {0:F2} mm/m2",
            averageRainInBurgas);
        PrintMonthsWithFilter(burgasArray);

        break;
    default:
        Console.WriteLine("Incorrect number of station!");
        break;
    }
}

static void InputArray(int[] arr)
{
    for (int i = 0; i < arr.Length; i++)
    {
        arr[i] = rand.Next(0, 201);
    }
}

static void OutputArray(int[] arr)
{
    for (int i = 0; i < arr.Length; i++)
    {
        Console.WriteLine("Raining in month " + (i + 1) + ": " + arr[i] + "
mm/m2");
    }
}

```

```

    }

    static double AverageAmountOfRainingForAYear(int[] arr)
    {
        double totalAmountOfRainForAYear = 0;

        for (int i = 0; i < arr.Length; i++)
        {
            totalAmountOfRainForAYear += arr[i];
        }

        return totalAmountOfRainForAYear / arr.Length;
    }

    static void PrintMonthsWithFilter(int[] arr)
    {
        bool thereAreMonths = false;

        for (int i = 0; i < arr.Length; i++)
        {
            if (arr[i] > 76 && arr[i] % 7 == 0)
            {
                thereAreMonths = true;
                Console.WriteLine("Month " + (i + 1) + " has amount of rain more than
76 mm/m2 and that amount can be divided by 7 without remainder.");
                break;
            }
        }

        if (!thereAreMonths)
        {
            Console.WriteLine("There aren't any months with rain more than 76
mm/m2, which can be divided by 7 without remainder.");
        }
    }
}

```

## Задача 6: SimpleArray

```
using System;
namespace SimpleArray
{
    class Program
    {
        static void Main(string[] args)
        {
            string command;
            Console.Write("Enter number of elements: ");
            int numberOfElements = int.Parse(Console.ReadLine());
            int[] simpleArray = new int[numberOfElements];
            InputArray(simpleArray);

            do
            {
                Console.WriteLine("Choose a command (swap, increase, decrease or info) or type 'End' if you want to exit:");
                command = Console.ReadLine();

                switch (command)
                {
                    case "swap":
                        SwapElementsInArray(simpleArray);
                        OutputArray(simpleArray);
                        break;
                    case "increase":
                        IncreaseArrayElementsUsingString(simpleArray);
                        OutputArray(simpleArray);
                        break;
                    case "decrease":
                        DecreaseArrayElementsUsingNumber(simpleArray);
                        OutputArray(simpleArray);
                        break;
                    case "info":
                        ArrayInfo(simpleArray);
                        OutputArray(simpleArray);
                        break;
                }
            } while (command != "End");

            OutputArray(simpleArray);
        }

        static void InputArray(int[] arr)
        {
            for (int i = 0; i < arr.Length; i++)
            {
                Console.Write("Number " + (i + 1) + ": ");
            }
        }
    }
}
```

```

        arr[i] = int.Parse(Console.ReadLine());
    }
}

static void OutputArray(int[] arr)
{
    for (int i = 0; i < arr.Length; i++)
    {
        Console.WriteLine("Number " + (i + 1) + ": " + arr[i]);
    }
}

static void SwapElementsInArray(int[] arr)
{
    int index1 = 0;
    int index2 = 0;
    int numberSwap = 0;

    bool index1IsNumber = false;
    bool index2IsNumber = false;

    string indexes;

    while (!index1IsNumber || !index2IsNumber)
    {
        Console.WriteLine("Enter the indexes of the two elements you want to
swap (seperate them with a space): ");
        indexes = Console.ReadLine();
        if (!indexes.Contains(" "))
        {
            Console.WriteLine("Input must contain only one space!!");
            continue;
        }

        index1IsNumber = int.TryParse(indexes.Substring(0, indexes.IndexOf(" ")),
out index1);
        index2IsNumber = int.TryParse(indexes.Substring(indexes.IndexOf(" ")),
out index2);

        if (index1 > arr.Length || index2 > arr.Length)
        {
            Console.WriteLine("The input numbers were greater than the length of
the array!");
            index1IsNumber = false;
            index2IsNumber = false;
        }
    }

    numberSwap = arr[index1];
    arr[index1] = arr[index2];

```

```

        arr[index2] = numberSwap;
    }

    static void IncreaseArrayElementsUsingString(int[] arr)
    {
        int charSum = 0;

        Console.WriteLine("Enter a word: ");
        string word = Console.ReadLine();

        foreach (char c in word)
        {
            Console.WriteLine($"{c} --> {(int)c}");
            charSum += c;
        }

        Console.WriteLine("Total sum: " + charSum);

        for (int i = 0; i < arr.Length; i++)
        {
            arr[i] += charSum;
        }
    }

    static void DecreaseArrayElementsUsingNumber(int[] arr)
    {
        bool inputIsNumber = false;
        int number = 0;

        while (!inputIsNumber)
        {
            Console.WriteLine("Enter a number: ");
            inputIsNumber = int.TryParse(Console.ReadLine(), out number);
        }

        for (int i = 0; i < arr.Length; i++)
        {
            arr[i] -= number;
        }
    }

    static void ArrayInfo(int[] arr)
    {
        int arraySum = 0;
        int greatestArrayElement = arr[0];
        int leastArrayElement = arr[0];

        for (int i = 0; i < arr.Length; i++)
        {
            if (arr[i] > greatestArrayElement)

```

```

    {
        greatestArrayElement = arr[i];
    }

    if (arr[i] < leastArrayElement)
    {
        leastArrayElement = arr[i];
    }

    arraySum += arr[i];
}

Console.WriteLine("Array count --> " + arr.Length);
Console.WriteLine("Max element --> " + greatestArrayElement);
Console.WriteLine("Min element --> " + leastArrayElement);
Console.WriteLine("Total sum --> " + arraySum);
}
}
}

```