# Практически проект: Познай числото

Logo

Description automatically generated with low confidence

Днес ще създадем конзолната игра **“Познай числото”**. Това е игра, в която вашият опонент (компютърът) избира **случайно число** между **1** и **100**. Вашата задача е да **познаете** кое число е избрал компютърът. След всяко число, което въведете, компютърът ще ви **насочи** дали числото е **по-голямо**, или **по-малко** от това, което той е избрал, докато не го **познаете**:

Text

Description automatically generated Text

Description automatically generated with medium confidence

## Създаване на GitHub репо

Вече имате създаден **GitHub** акаунт, така че можете директно да създадете ново **репо (хранилище)**.

Можете да го създадете от следния линк: <https://github.com/new>. Изберете **смислено име**, например

"GuessANumberBy{вашето потребителско име}", добавете **кратко описание** и направете репото **публично**:

Graphical user interface, text, application

Description automatically generated

|  |  |
| --- | --- |
| Icon  Description automatically generated | Моля изберете **оригинално име** за вашия проект!  Вашият GitHub профил трябва да бъде **уникален**, а не същият като тези на съучениците ви.  Можете да следвате този документ, но е позволено да **правите промени** и да **добавяте свои функционалности**. |

**Добавете и** README.md файл и .gitignore **за Visual Studio**, както е показано:

Graphical user interface, text, application, email

Description automatically generated

В Git проектите **файлът** .gitignore уточнява кои файлове от репото **не са част** от сорс кода и трябва да бъдат **игнорирани**.

**Променете лиценза** на “**MIT**” и кликнете **бутона** [Create], за да **създадете репото**:

Graphical user interface, text, application, email

Description automatically generated

Сега вашето **репо е създадено** и изглежда по подобен начин:

Graphical user interface, text, email, website

Description automatically generated

## Код за играта

### Създаване на проект във Visual Studio

Първо, трябва да **стартираме Visual Studio** и да **създадем нов C# console application**. След това трябва да **изберем подходящо име** и **място за съхранение** на проекта. На следващия екран изберете [.NET 6 (Long-term support)] и създайте проекта.

Graphical user interface, text, application

Description automatically generated

Преди да продължим, нека да променим името на главния клас - Program.cs – на нещо **по-смислено**. Вече знаете как да го направите:

Graphical user interface, text, application

Description automatically generated

### Имплементиране на логиката на играта

#### Генериране на избор на компютъра

Създайте променлива от тип "Random", който вече сте учили. Тази променлива ще ни помогне да **изберем случайно число**, използвайки метода "Next()". Ще го използваме, за да може компютърът да избира случайно число **между 1 и 100** всеки път.

A screenshot of a computer

Description automatically generated with medium confidence

Повече информация за Random.Next(): <https://docs.microsoft.com/en-us/dotnet/api/system.random.next?view=net-6.0> .

#### Прочитане на входа на потребителя

Сега можем да напишем while цикъл, за да итерираме докато потребителят **познае** **случайното число**. Напишете на конзолата какво трябва да направи потребителят и **прочетете входните данни**. Вече знаете как да постигнете това:

Graphical user interface, text

Description automatically generated with medium confidence

Сега нека да изпълним **приложението** в конзолата и да проверим дали текущият код **работи правилно**:

Graphical user interface

Description automatically generatedText

Description automatically generated

Можем да видим, че нашият текст **е изписан на конзолата** и трябва да **прочетем** входа на потребителя **повторно**, защото ползваме while цикъл.

#### Проверка на входа на потребителя

Сега **проверете** входа на потребителя, използвайки метода int.TryParse(). Това ще тества входните данни и ще върне "true" или "false" в зависимост от въведения от потребителя **вход**. Ако е **число** (**както се очаква**", методът ще върне "true", а ако не е число - "false".

Като **втори параметър** на метода ще даден **типа** и **името** на **променливата**, в която ще **съхраним** избора на потребителя, ако е въвел **валидни данни**:

Graphical user interface, text

Description automatically generated

Малко повече информация за int.TryParse(): <https://docs.microsoft.com/en-us/dotnet/api/system.int32.tryparse?view=net-6.0> .

Сега имаме създадена Boolean променлива"isValid" и можем да напишем if-else конструкция. Първо, нека да проверим дали входните данни са **валидни**:

Background pattern

Description automatically generated with low confidence

Ако данните са валидни, напишете **вложена** if-else конструкция, в която да проверите **трите възможни варианта**.

Първо, ако числото, което потребителят е избрал, **е равно** на това, което компютърът е генерирал, потребителят **е познал** числото. **Отпечатайте** съобщение и **прекратете** изпълнението на цикъла чрез ключовата дума "break":

Graphical user interface, text, application, email, website

Description automatically generated

Другите **два случая** са когато изборът на потребителя е **по-голям** или **по-малък** от този на компютъра. Напишете останалата част от конструкцията сами:

Graphical user interface, text, application

Description automatically generated

Сега нека да изпълним **приложението** и да проверим дали кодът **работи правилно**:

Graphical user interface

Description automatically generatedText

Description automatically generated with medium confidence

Можем да видим, че приложението работи добре, но още не сме приключили.

#### Проверка за невалиден вход

Сега трябва да довършим нашия **else** блок за **финалния** случай, в който входа на потребителя е **невалиден**.

A picture containing graphical user interface

Description automatically generated

Целият ви код трябва да изглежда по подобен начин:

Graphical user interface, text, application, email

Description automatically generated

#### Тестване на приложението

След като го **стартирате**, приложението трябва да изглежда така:

Graphical user interface, text

Description automatically generated Graphical user interface, text

Description automatically generated

Text

Description automatically generated Text

Description automatically generated

Text

Description automatically generated

## Качване на проекта в GitHub

Вече знаете как да качите проекта с **Git** **Bash** или **TortoiseGit**.

### Използвайки TortoiseGit (Опция 1)

Използвайте **Git clone** за клониране с TortoiseGit. Отидете в директорията, в която искате да го съхраните, натиснете **десен бутон** в папката и изберете click [Git Clone**]**.

Graphical user interface, application

Description automatically generated

Сега отидете във вашия **GitHub** профил, отворете новосъздаденото **репо** и копирайте неговия **repo URL**.

Отворете TortoiseGit, поставете **URL**-а и кликнете [OK].

Graphical user interface, application

Description automatically generated

Файловете от вашето GitHub репо ще бъдат свалени в **подпапка** с името на проекта в GitHub, в случая " **GuessANumberByPeter** ".

Graphical user interface, text, application, email

Description automatically generated

Когато отворим папката, която съдържа клонираното репо, трябва да изглежда така:

Graphical user interface

Description automatically generated with low confidence

Следващото нещо, което трябва да направите, е да **добавите проекта** към **клонираното репо**. Можете да използвате “**Cut & Paste**”. Трябва да изглежда така:

Graphical user interface, text, application

Description automatically generated

Сега нека **качим** промените на проекта в GitHub.

Можем да използваме [GitCommit…]от TortoiseGit. Отидете в папката на проекта, натиснете **десен бутон** и кликнете[Git Commit -> "main"…].

Graphical user interface, text, application

Description automatically generated

Добавете **подходящо** съобщение и кликнете [Add], за да не пропуснете някои файлове, и след това изберете [Commit].

Graphical user interface, application

Description automatically generated

След това кликнете [Push] и после [OK].

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Това е всичко, от което се нуждаете, за да **качите вашия проект** във вашето **GitHub репо**, използвайки TortoiseGit.

### Използвайки Git Bash (Опция 2)

Като **алтернатива на предишния вариант**, ако нямате "**TortoiseGit**", можете да използвате"**Git Bash**", за да качите проекта си във вашето GitHub репо.

Първо, ако нямата Git на своя **компютър**, го **инсталирайте** от <https://git-scm.com/downloads>.

Отидете в **директорията** с проекта, натиснете **десен бутон** в папката и изберете "**Git Bash Here**", за да отворите конзолата на Git. Ако менюто "**Git Bash Here**" **липсва**, първо трябва да инсталирате Git.

Graphical user interface, application

Description automatically generated

Напишете командата **"**gitclone**"**, последвана от линка на **репото**:

|  |
| --- |
| **git clone** |

Поставете **URL**-а към вашето репо след командата.

Text

Description automatically generated

Файловете от вашето репо ще бъдат свалени в **подпапка** с името на самия проект в GitHub, в нашия случай "**GuessANumberByPeter**".

Graphical user interface, text, application, email

Description automatically generated

Когато отворите папката, тя трябва да изглежда така:

Graphical user interface

Description automatically generated with low confidence

Следващото нещо, което трябва да направите, е да **добавите файловете от проекта** в **клонираната папка**. Трябва да изглежда така:

Graphical user interface, text, application

Description automatically generated

Сега сме готови да качим нашите промени. Отидете в папката на репото, натиснете **десен бутон**, изберете"GitBashHere" и изпълнете следните **команди**.

|  |
| --- |
| git status |

**Git status** показва състоянието на директорията.

A screenshot of a computer

Description automatically generated

Сега напишете:

|  |
| --- |
| git add . |

Горната команда **добавя** всички нови/модифицирани файлове към вашето локално **Git repo**.

Сега напишете:

|  |
| --- |
| git commit -m "Uploaded the game Guess a number" |

Тази команда **commit**-ва вашите промени във вашето локално **Git repo**. Също трябва да **добавите** подходящо **съобщение**.

Имаме да изпълним **още две** **команди**. Напишете:

|  |
| --- |
| git pull |

Тази команда **обновява** вашето локално **репо** от GitHub. Сваля последната версия на проекта от GitHub и я слива с вашето локално копие.

Последното нещо, което трябва да направим, е да **push**-нем промените чрез следната команда:

|  |
| --- |
| git push |

Тази команда **качва вашите локални промени в GitHub**.

Text, chat or text message

Description automatically generated

Това е всичко, от което се нуждаете, за да **обновите** вашето **репо**, използвайки **Git Bash**.

Можете да прочетете повече за Git Bash тук: <https://git-scm.com/about>.

## \* Модифициране на кода, добавяне на нови функционалности

Нека да си **поиграем с кода** и **да го модифицираме**.

Below are a few **ideas** of what you can implement or modify as an addition to your code.

|  |  |
| --- | --- |
| Icon  Description automatically generated | Това е ваш собствен проект. **Бъдете уникални**.   * Имплементирайте **свои собствени функционалности**. * **Имплементирайте кода сами**, добавете ваш стил, форматиране, коментари и т.н. |

### Добавете трудност

Добавете логика за трудност, така че потребителят да има **само няколко опита** да познае числото.

### Рестартиране на играта

Можете **автоматично да рестартирате играта**, след като е приключила, или да попитате потребителя дали иска да играе отново.

### Допълнителни идеи

* Можете да **добавите нива**, така че всеки път, когато потребителят познае числото, диапазонът да се увеличава, напр. **Ниво 1 (1 - 100)**, **Ниво 2 (1-200)** и т.н.

### Качване в GitHub

**Сега качете промените в кода си** във вашето GitHub репо!

|  |  |
| --- | --- |
| Icon  Description automatically generated | Много е важно да **commit-вате често** вашия код в GitHub. По този начин създавате **богата commit история** за вашия проект:  A picture containing chart  Description automatically generated |

## Създаване на README.md файл

Силно препоръчително е да напишете **документация за вашия проект в GitHub**, за да опишете за какво е предназначен. Започнете като редактирате файла README.md от вашето репо:

Graphical user interface, text, application, email

Description automatically generated

Добавете име на проекта. Използвайте "#" пред текста, за да индикирате **заглавието**:

Graphical user interface, application

Description automatically generated

Можете да **видите** прогреса си, като натиснете бутона [Preview]:

### Секции на документацията

Добавете **информация** за вашия проект във вашия README.md файл. Обикновено документацията съдържа следните **секции**:

* **Заглавие на проекта** (трябва да отговаря на въпроса "Какво има в проекта?")
* **Цели на проекта**  (какъв проблем решаваме, напр. имплементация на игра)
* **Решение**  (трябва да описва как решаваме проблема 🡪 **алгоритми**, **технологии**, **библиотеки** и т.н.)
* **Линк към source код**
* **Скрийншоти** (скрийншоти към различните функционалности на проекта)
* **Демо наживо** (линк към директно демо на проекта)

### Използвайте markdown

Забележете, че README.md файла се пише на **Markdown език**. Markdown комбинира текст и специално форматирани тагове.

Можете да научите повече за **Markdown** тук: <https://docs.github.com/en/get-started/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax>.

### Цели на проекта

Започнете документацията, като опишете **целите на проекта**. Какъв проблем решава проекта?

### Примерна документация

Това е **пример** как може да документирате проекта си. Съветваме ви да не го преписвате, а сами да структурирате документацията си!

Graphical user interface, text, application, email

Description automatically generated

Можете да добавите **подходящи изображения**, за да подобрите документацията. Можете да добавите изображение по следния начин:



Добавете информация за **входа** и **изхода** на проекта:

Graphical user interface

Description automatically generated with medium confidence

### Вашето решение

Опишете как **решавате** проблема: **алгоритми, технологии, библиотеки и т.н.**

### Линк към сорс код

Добавете **линк** към вашия **сорс код**, както следва:

Logo

Description automatically generated with low confidence

### Скрийншоти

Добавете **скрийншоти** на вашия проект:

1. **Направете скрийншот** с предназначен за това инструмент (например [Snipping Tool](https://support.microsoft.com/en-us/windows/open-snipping-tool-and-take-a-screenshot-a35ac9ff-4a58-24c9-3253-f12bac9f9d44) за Windows).
2. **Поставете** скрийншота във файла за документация, използвайки [Ctrl+V]:

Примерни скрийншоти за играта "Guess a Number":

Text

Description automatically generated

## Качване на приложението си в Replit

Вече би трябвало да имате **Replit** профил. Добавете вашия **проект**, за да можете да показвате **проектите си** на вашите приятели и да сложите линка в секцията "**live demo links**" във вашата GitHub документация.

Създайте **нов проект** в Replit от **менюто** в **горния** **ляв ъгъл**.

Кликнете [Create], след това избере **езика**, на който е **написан** вашият проект, изберете **име** и **създайте** проекта. Изберете "Mono C#".

Graphical user interface, text, application, chat or text message

Description automatically generated

**Поставете вашия код** във файла "Main.cs":

.

Graphical user interface, text, application

Description automatically generated

**Кликнете** [Run] и ще изпълните приложението си директно в браузъра:

Text

Description automatically generated

## Добавете Replit линк към вашия README.md файл

Сега добавете линк към демо от вашия проект към вашата GitHub документацията:

A picture containing text

Description automatically generated

Можете да направите **скрийншот** от Replit.com и **да го поставите** в GitHub документацията директно чрез **[Ctrl+V]**.

След промените, документацията трябва да изглежда така:

Graphical user interface, text, application

Description automatically generated

Поздравления! Успешно завършихте вашата **втора конзолна игра** и вашия **втори проект** в **GitHub портфолиото ви**.