# Лаб: Повторения с цикли – while-цикъл

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

**Тествайте** решението си в **judge системата**: <https://judge.softuni.bg/Contests/Compete/Index/1014>.

## Число в диапазона [1…100]

Напишете програма, която въвежда цяло положително **число** n **в диапазона [1…100]**. При въвеждане на число извън посочения диапазон, да се отпечата съобщение за грешка и **да се въведе ново число**.

### Примерен вход и изход

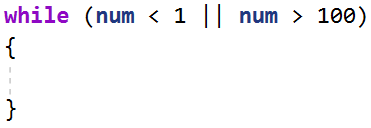
|  |
| --- |
| **Вход / Изход** |
| **35**  The number is: 35 |
| **105**  Invalid number!  **0**  Invalid number!  **-200**  Invalid number!  **77**  The number is: 77 |

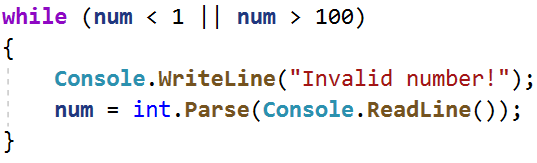
### Насоки

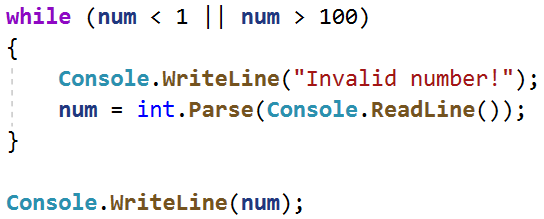
1. Създайте **нов проект** в съществуващото Visual Studio решение с име **"Number-In-Range"**.
2. **Прочетете входните данни от конзолата в искания вид – едно цяло число number.**



1. Направете един **while** цикъл, който да се повтаря **докато** съответното число number **е по-малко от 1 или по-голямо от 100.**



1. Ако програмата **влезе в тялото на цикъла**, то това ще означава, че потребителят е въвел число, което не е в диапазона [1...100]. В тялото на цикъла първо печатайте "Invalid number!", след което **прочетете ново число** и **присвоете стойността му на променливата** number**.**
2. **След тялото на цикъла** отпечатайте променливата num в желания формат. Тя ще има стойност **между 1 и 100**, понеже програмата ще е **излязла от цикъла.**



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

## Парола

Напишете програма, която първоначално прочита име и парола на потребителски профил. След това чете парола за вход, при въвеждане на грешна парола, потребителя да се подкани да въведе нова парола.

### Примерен вход и изход

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |
| Nakov  1234  pass  1324  1234 | Welcome Nakov! | Gosho  secret  secret | Welcome Gosho! |

### Насоки

1. Инициализирайте две променливи username и password, които ще съдържат потребителското име и паролата:



1. Инициализирайте променлива input, която ще държи въведената от потребителя парола за вход:



1. В while цикъл, до въвеждане на валидна парола, четете нова:



1. Когато се въведе **валидна парола** **принтирайте съобщението за успешен вход**:



## Редица числа 2k+1

Напишете програма, която чете число n, въведено от потребителя,и отпечатва **всички числа ≤ n от редицата**: 1, 3, 7, 15, 31, …. Всяко следващо число се изчислява като умножим **предишното** с **2** и добавим **1**.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 3 | 1  3 | 8 | 1  3  7 | 17 | 1  3  7  15 | 31 | 1  3  7  15  31 |

### Насоки

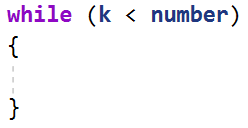
1. Прочетете от конзолата цяло число.



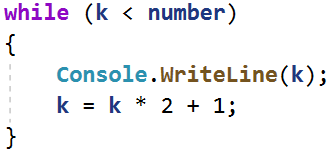
1. **Създайте** променлива от тип **цяло число**, която ще е брояч и има **първоначална стойност 1**.



1. **Създайте while** цикъл, който се повтаря докато **брояча е по-малък** от числото, което сте прочели от конзолата.



1. При всяко повторение на цикъла **принтирайте стойността на брояча** и му **прибавяйте дадената стойност**.



## Баланс по сметка

Напишете програма, която пресмята колко общо пари има в сметката, след като направите определен брой вноски. На първия ред ще получите **колко вноски трябва да се направят**. На всеки следващ ред ще получавате сумата, която трябва да внесете в сметката, **докато не се достигне броя вноски**. При всяка получена сума на конзолата трябва да се извежда **"Increase: "** + сумата и тя да се **прибавя в сметката**. Ако получите число **по-малко от 0** на конзолата трябва да се изведе **"Invalid operation!"** и **програмата да приключи**. Когато програмата приключи трябва да се принтира **"Total: "** + общата сума в сметката закръглена до втория знак след десетичната запетая.

### Примерен вход и изход

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |
| 3  5.51  69.42  100 | Increase: 5.51  Increase: 69.42  Increase: 100.00  Total: 174.93 | 5  120  45.55  -150  55 | Increase: 120  Increase: 45.55  Invalid operation!  Total: 165.55 |

### Насоки

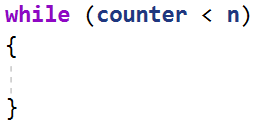
1. Четете число от конзолата, което определя колко вноски трябва да се направят.



1. Създайте **две променливи** - една променлива, която ще е брояч и една, в която да пазите сумата, която е внесена.



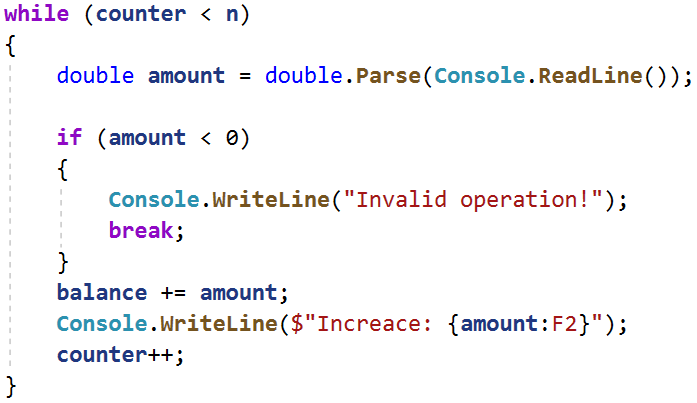
1. Създайте **while** цикъл, който ще продължава докато брояча е по-малък от броя вноски.



1. При всяко повтаряне на цикъла, ще четете сума от конзолата **(реално число)**.



1. Създайтепроверка, **която проверява дали е въведена отрицателна сума** и ако е така принтирайте на конзолата нужното съобщение и прекратявайте цикъла, ако сумата е положителна, **увеличавайте баланса по сметката**, принтирайте на конзолата с колко сте увеличили баланса и **увеличавайте брояча с едно.**



1. След цикъла принтирайте тоталния баланс.



## Най-голямо число

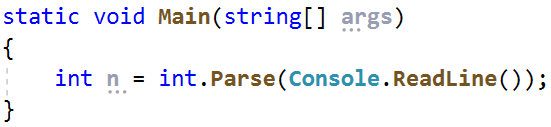
Напишете програма, която чете n-на брой **цели числа** (n > 0), въведени от потребителя, и намира **най-голямото** измежду тях. Първо се въвежда броят числа n, а след това самите n числа, по едно на ред.

### Примерен вход и изход

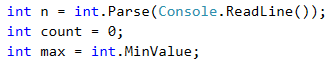
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 2  100  99 | 100 | 3  -10  20  -30 | 20 | 4  45  -20  7  99 | 99 | 1  999 | 999 | 2  -1  -2 | -1 |

### Насоки

1. Първо въведете едно число n (броят числа, които предстои да бъдат въведени).

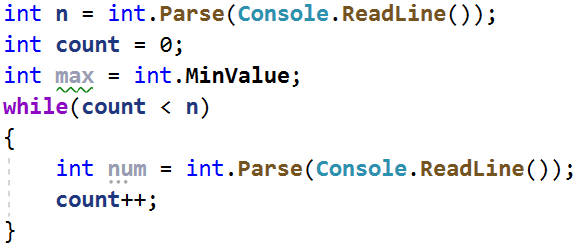


1. Създайте променливите **count,** която ще брои колко числа сме прочелииmax като за стойност сложете **минималната стойност на** int **като използвате** int.MinValue

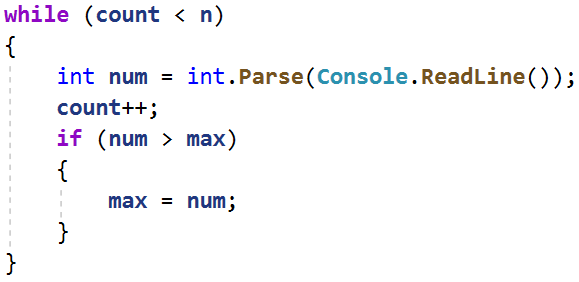


Повече информация за обхвата на типовете данни можете да прочетете [тук](https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/types-and-variables)

1. Направете цикъл до **n** и на който четете число num и увеличава брояча с 1**.**



1. Направете **проверка** дали числотоnum **е по-голямо от** max и ако е, приравнете стойността на max до num



1. Накрая отпечатайте най-голямото число



## Най-малко число

Напишете програма, която чете n-на брой **цели числа** (n > 0), въведени от потребителя и намира **най-малкото** измежду тях. Първо се въвежда броят числа n, а след това самите n числа, по едно на ред.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 2  100  99 | 99 | 3  -10  20  -30 | -30 | 4  45  -20  7  99 | -20 | 1  999 | 999 | 2  -1  -2 | -2 |

### Насоки

1. Задачата е аналогична на предходната.
2. Използвайте за първоначална стойност на min = int.MaxValue() – най-голямата стойност на типа int

## Завършване

Напишете програма, която изчислява **средната оценка** на ученик от **цялото му обучение**. На първия ред ще получите **името на ученика**, а на всеки следващ ред неговите годишни оценки. Ученикът преминава в следващия клас, **ако годишната му оценка е** **по-голяма или равна на 4.00**. **Ако оценката му е под 4.00**, той ще повтори класа.

При успешно завършване на **12-ти** клас да се отпечата:

"{**име на ученика**} graduated. Average grade: {**средната оценка от цялото обучение**}"

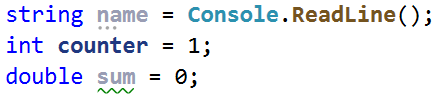
**Стойността трябва да бъде форматирана до втория знак** след десетичната запетая.

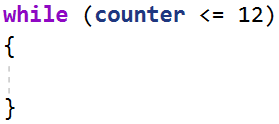
### Примерен вход и изход

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вход** | **изход** |  | **вход** | **изход** |
| Pesho  3.5  4  5.5  6  5.43  4.5  6  5.55  5  6  6  5.43  5 | Pesho graduated. Average grade: 5.37 | Ani  5  5.32  6  5.43  5  6  5.5  4.55  5  6  5.56  6 | Ani graduated. Average grade: 5.45 |

### Насоки

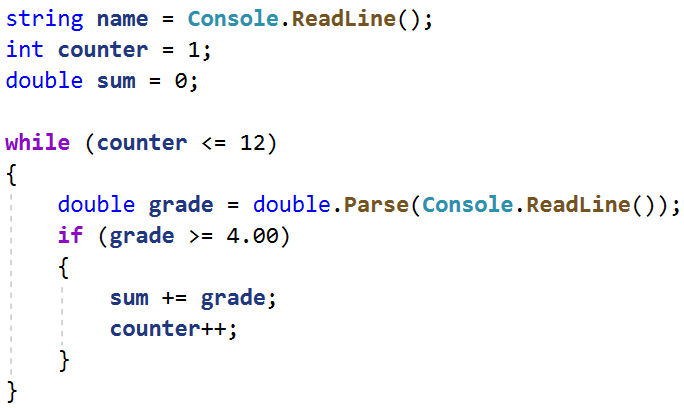
1. Прочетете **едно име** от конзолата. Създайте **две помощни променливи** – една променлива **брояч** **с начална стойност 1** и една променлива с начална стойност 0, в която ще държим **сумата от всички оценки** на дадения ученик.



1. Направете **един** while **цикъл**, който да се повтаря **докато брояча е по-малък или равен на 12**. 
2. **В тялото на цикъла четете оценката на ученика за съответния клас (реално число в интервала [2.00...6.00]).**



1. След това направете **една проверка дали въведената оценка е по-голяма от 4.00** и **ако е**, **я добавете към помощната променлива за сума**, която направихте в началото. **Увеличете** **стойността на брояча с 1**, понеже **щом оценката е 4 или повече**, **то ученикът е преминал в следващия клас**.



1. **След тялото на** while **цикъла**, изчислете **средната оценка** **от цялото обучение** като разделите **сумата от всички оценка на 12.**



1. **Принтирайте** резултата в желания формат.



## Завършване - част 2

Напишете програма, която изчислява **средната оценка** на ученик от цялото му обучение. На първия ред ще получите **името на ученика**, а на всеки следващ ред неговите годишни оценки. Ученикът преминава в следващия клас, **ако годишната му оценка е по-голяма или равна на 4.00**. **Ако ученикът бъде скъсан повече от един път, то той бива изключен и програмата приключва,** като се отпечатва **името на ученика и в кой клас бива изключен**.

При успешно завършване на **12-ти** клас да се отпечата :

"{**име на ученика**} graduated. Average grade: {**средната оценка от цялото обучение**}"

**В случай, че ученикът е изключен от училище, да се отпечата:**

"{**име на ученика**} has been excluded at {**класа, в който е бил изключен**} grade"

**Стойността трябва да бъде форматирана до втория знак** след десетичната запетая.

### Примерен вход и изход

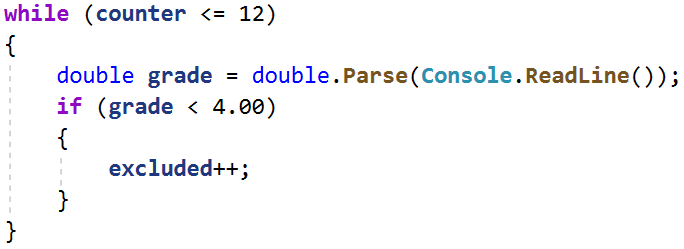
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |
| Gosho  5  5.5  6  5.43  5.5  6  5.55  5  6  6  5.43  5 | Gosho graduated. Average grade: 5.53 | Mimi  5  6  5  6  5  6  6  2  3 | Mimi has been excluded at 8 grade |

### Насоки

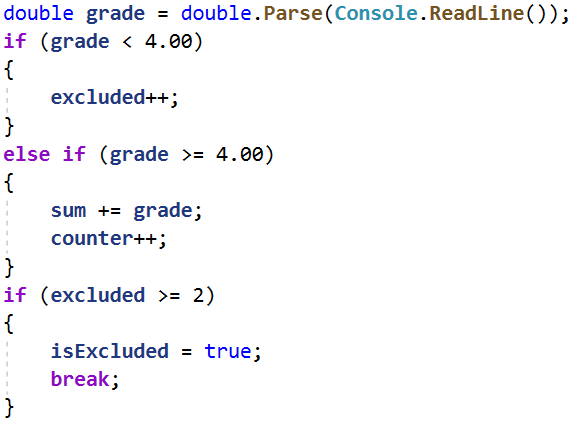
1. Използвайте решението от предната задача и добавете нужната функционалност:
2. Направете **още две помощни променливи в началото**, които да следят **дали ученика трябва да бъде изключен**. Едната променлива ще е **брояч** и трябва да е **от тип цяло число** и с **първоначална стойност нула**. С нея ще следим **колко пъти ученикът е трябвало да повтаря**. Другата променлива трябва да е от **булев тип** и да е **с началната стойност false**.



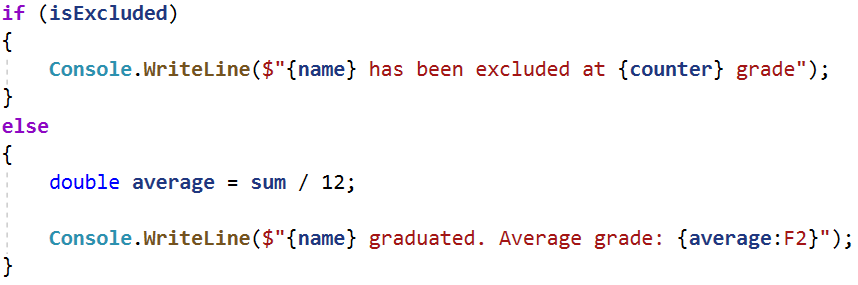
1. **В тялото на while цикъла** проверявайте **дали съответната оценка е по-ниска от 4.00** и ако е, **увеличавайте брояча с 1**.



1. Направете **още една проверка дали ученикът е бил изключван 2 пъти** (като използвате съответния брояч) и ако е, **променете стойността на булевата променлива на true и излезте от цикъла**.



1. След тялото на цикъла, **печатайте изхода в зависимост от това, дали ученикът е бил изключен или не**.



# Примерна изпитна задача

## \*Преместване

На осемнадесетия си рожден ден на Хосе взел решение, че ще се изнесе да живее на квартира. Опаковал багажа си в **кашони** и намерил подходяща обява за апартамент под наем. Той започва да пренася своя багаж **на части**, защото не може да пренесе целия наведнъж. Има ограничено **свободно пространство** в новото си жилище, където може да разположи вещите, така че мястото да бъде подходящо за живеене.

Напишете **програма, която изчислява свободния обем от жилището на Хосе, който остава след като пренесе багажа си.**

**Бележка: Един кашон е с точни размери: 1m. x 1m. x 1m.**

### Вход

Потребителят въвежда следните данни на отделни редове:

1. **Широчина на свободното пространство - цяло число в интервала [1...1000]**
2. **Дължина на свободното пространство - цяло число в интервала [1...1000]**
3. **Височина на свободното пространство - цяло число в интервала [1...1000]**
4. На следващите редове **(до получаване на команда "Done") -** **брой кашони, които се пренасят в квартирата - цели числа в интервала [1...10000];**

**Програмата трябва да приключи прочитането на данни при команда "Done"** **или ако свободното място свърши.**

### Изход

Да се **отпечата** на конзолата **един** от следните редове:

* Има още свободно място:

**"{брой свободни куб. метри}** **Cubic meters left."**

* Ако свободното място свърши преди да е дошла команда **"**Done**":**

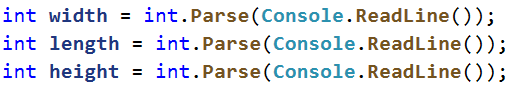
**"****No more free space! You need {брой недостигащи куб. метри} Cubic meters more."**

**Примерен вход и изход**

|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснение** |
| 10  10  2  20  20  20  20  119  100  200 | No more free space! You need 2 Cubic meters more. | 10 \* 10 \* 2 = **200 кубични метра.** 20 + 20 + 20 + 20 + 122 = **202** **кубични метра.** 200 - 202 = **2 недостигащи** кубични метра |
| 10  1  2  4  6  **Done** | 10 Cubic meters left. | 10 \* 1 \* 2 = **20 кубични метра.** 4 + 6 = **10** **кубични метра.** 20 - 10 = **10 кубични метра.** |

### Насоки

1. Прочетете три цели числа от конзолата, които са **широчина, дължина и височина**.



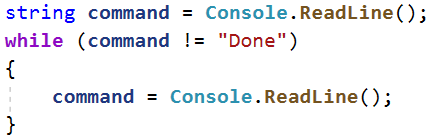
1. Пресметнете **колко кубични метра е апартамента**.



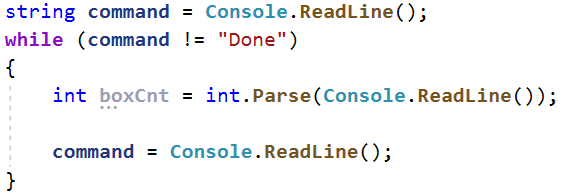
1. Създайтедве променливи, една за **пространството, което заемат кашоните** и една, в която ще запазите **разликата между пространството, което заемат кашоните и пространството на апартамента**.



1. Създайте **while** цикъл, който се повтаря до получава на команда **DONE** и при всяко повторение се прочита един ред от конзолата.



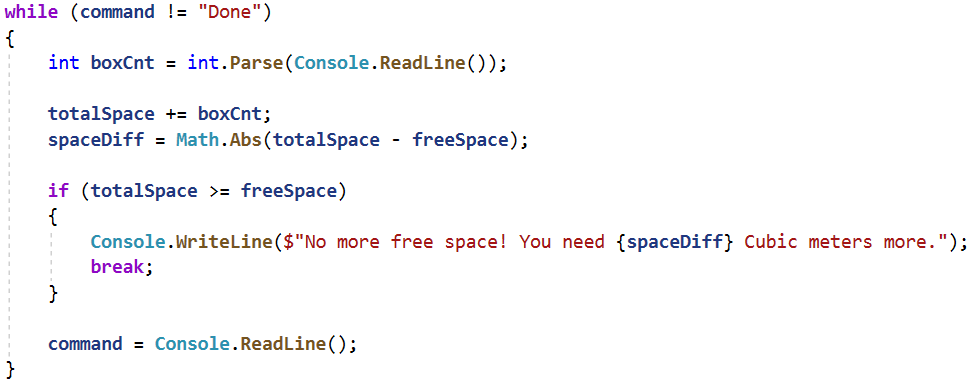
1. Ако командата не е **DONE** променете типа на командата в **цяло число**.



1. Добавете пространството, което заемат внесените кашони към по-рано създадената променлива и пресметнете разликата.



1. Ако **напълните или препълните апартамента**, принтирайте на конзолата съобщение и да **спрете цикъла.**



1. Ако цикъла приключи с команда **DONE** принтирайте на конзолата съобщението.

