# Лаб: Вложени цикли

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

**Тествайте** решенията си в **Judge системата**: [judge.softuni.bg/Contests/Compete/Index/1016](https://judge.softuni.bg/Contests/Compete/Index/1016)

## Числата от N до 1 в обратен ред

Напишете програма, която чете цяло положително число n, въведено от потребителя, и печата **числата от** n **до** **1** **в обратен ред** (от най-голямото към най-малкото).

### Примерен вход и изход

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 2 | 2  1 | 3 | 3  2  1 | 5 | 5  4  3  2  1 |

### Насоки:

1. Извадете **едно число** от input – **n:**



1. Направете **един for цикъл от** n **до 0**, но **вместо да увеличавате** **променливата** i с 1, я **намаляйте** **с 1** при **всяка итерация на цикъла**:



1. **В тялото на цикъла** **принтирайте променливата** i:



## Числата от 1 до N през 3

Напишете програма, която чете число n, въведено от потребителя, и отпечатва **числата от 1 до** n **през 3** (със стъпка 3).

### Примерен вход и изход

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 10 | 1  4  7  10 | 7 | 1  4  7 | 15 | 1  4  7  10  13 |

### Насоки:

1. Вземете елемент от аргумента на функцията – **едно число n**



1. Направете **for цикъл** **от 1 до n** (**включително**) и задайте **стъпка 3**. Това означава, че при всяка итерация на цикъла, променливата **i** **ще увеличава стойността си с 3, вместо с 1:**

 **В тялото на цикъла** **принтирайте променливата** **i**:



## Четни степени на 2

Да се напише програма, която чете число n, въведено от потребителя, и **печата четните степени на 2** **≤** **2n**: **20**, **22**, **24**, **28**, …, **2n**.

### Примерен вход и изход

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |  | **вход** | **изход** |
| 3 | 1  4 | 4 | 1  4  16 | 5 | 1  4  16 | 6 | 1  4  16  64 | 7 | 1  4  16  64 |

### Насоки:

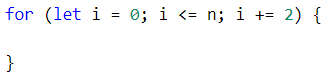
1. Вземете от аргумента – **едно число n:**



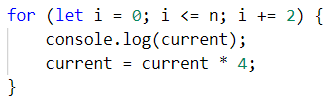
1. **Инициализирайте** помощна **променлива** и **й задайте първоначална стойност 1** (Ще я използваме, за да печатаме една степен на числото 2 при всяка итерация на цикъла).



1. Направете **for цикъл от 0 до n (включително)** и за да отпечатате **само четните степени на 2** му задайте **стъпка 2**:



1. В **тялото на цикъла** **принтирайте** помощната **променлива**, която **създадохме**, след което я **умножавайте по 4:**



## Комбинации

Напишете програма, която изчислява **колко решения в естествените числа** (включително и нулата) има уравнението:

**x1 + x2 + x3 + x4 + x5 = n**

**Числото n е цяло число и се въвежда от конзолата.**

### Примерен вход и изход

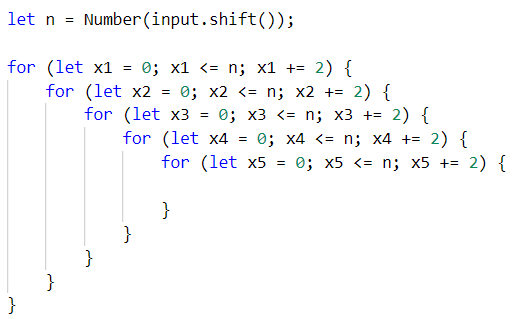
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** | **Вход** | **Изход** | **Вход** | **Изход** |
| 25 | 23751 | Генерираме всички комбинации от 5 числа, като първата е:  0+0+0+0+0=0, но понеже не е равна на 25, продължаваме:  0+0+0+0+1=1 – също не е 25 и т.н  Стигаме до първата валидна комбинация:  0 + 0 + 0 + 0 + 25 = 25, увеличаваме броя на валидни комбинации с 1,втората валидна комбинация е:  0 + 0 + 0 + 1 + 24 = 25  Третата:  0 + 0 + 0 + 2 + 23 = 25 и т.н  След генериране на всички възможни комбинации, броят на валидните е 23751. | 20 | 10626 | 5 | 126 |

### Насоки:

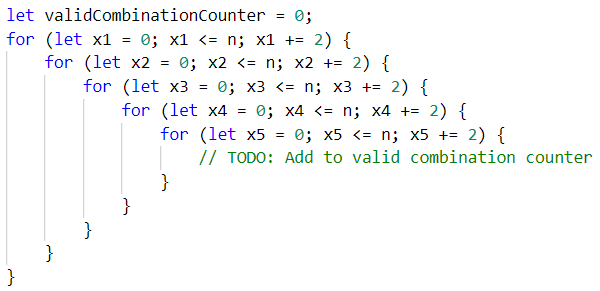
1. Извадете входните данни – едно число и го запаметете в променлива:



1. Създайте 5 вложени for-цикъла, с които да итерирате всяка възможна стойност на едно от 5те числа в уравнението:



1. Направете проверка в най-вътрешния вложен цикъл за стойностите на x1, x2, x3, x4, x5 във всяка една итерация. За да бъде валидно уравнението техният сбор трябва да е равен на n. Създайте променлива validCombinationsCount, която да пази броя на валидните комбинации и добавяйте към нея всеки път, когато генерирате такава:



1. Накрая принтирайте броя на валидните комбинации (validCombinationsCount).

## Сграда

Напишете програма, която извежда на конзолата номерата на стаите в една сграда (в низходящ ред), като са изпълнени следните условия:

* На **всеки четен етаж има само офиси**
* На **всеки нечетен етаж има само апартаменти**
* Всеки **апартамент** се означава по следния начин : **А**{**номер на етажа**}{**номер на апартамента**}, **номерата на апартаментите започват от 0.**
* Всеки **офис** се означава по следния начин : **О**{**номер на етажа**}{**номер на офиса**}, **номерата на** **офисите** **също започват от 0.**
* **На последният етаж винаги има апартаменти** и те са по-големи от останалите, за това **пред номера им пише 'L', вместо 'А'. Ако има само един етаж, то има само големи апартаменти!**

От конзолата се прочитат две **цели числа - броят на етажите и броят на стаите за един етаж.**

### Примерен вход и изход

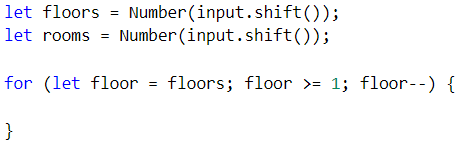
|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** | |
| 6  4 | L60 L61 L62 L63  A50 A51 A52 A53  O40 O41 O42 O43  A30 A31 A32 A33  O20 O21 O22 O23  A10 A11 A12 A13 | Имаме общо 6 етажа, с по 4 стаи на етаж. Нечетните етажи имат само апартаменти, а четните само офиси. | |
| **Вход** | **Изход** | **Вход** | **Изход** |
| 9  5 | L90 L91 L92 L93 L94  O80 O81 O82 O83 O84  A70 A71 A72 A73 A74  O60 O61 O62 O63 O64  A50 A51 A52 A53 A54  O40 O41 O42 O43 O44  A30 A31 A32 A33 A34  O20 O21 O22 O23 O24  A10 A11 A12 A13 A14 | 4  4 | L40 L41 L42 L43  A30 A31 A32 A33  O20 O21 O22 O23  A10 A11 A12 A13 |

### Насоки:

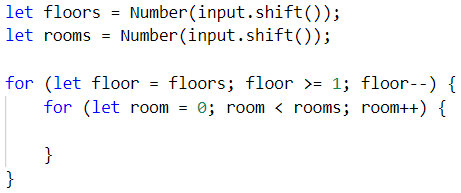
1. Извадете от аргумента – **цели числа**, които представляват **броя на етажите** и **стаите** на всеки един етаж и ги запазете в променливи:



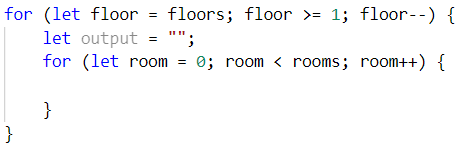
1. Създайте for-цикъл, който да итерира **броят на етажите от сградата в низходящ ред**:



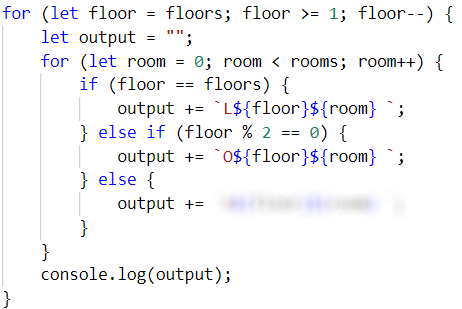
1. Създайте вложен for-цикъл, който да итерира **броят на стаите за всеки етаж**:



1. Създайте една помощна променлива, в която ще добавяте всяка стая от етажа. Инициализирайте я във външния цикъл, за да може при всяка негова итерация тя да бъде създавана отново.



1. Извършете множество проверки за **номера на етажа от итерацията на цикъла** и **добавяйте в променливата, която създадохте по-рано.**



1. Отпечатвайте всеки път, когато вътрешния цикъл направи необходимия брой итерации.

## Пътуване

Ани обича да пътува и иска тази година да посети **няколко** различни дестинации. Като си избере дестинация, ще прецени **колко пари ще й трябват**, за да отиде до там и ще започне да **спестява**. Когато е спестила **достатъчно**, ще може да пътува.

От **конзолата всеки път ще се четат първо дестинацията и минималния бюджет**, който ще е нужен за пътуването.

След това ще се четат **няколко суми**, които Ани спестява като работи и **когато успее да събере достатъчно за пътуването, ще заминава, като на конзолата трябва да се изпише:**

**"Going to {дестинацията}!"**

Когато е посетила всички дестинации, които иска, **вместо дестинация ще въведе "End"** и програмата ще приключи.

### Примерен вход и изход

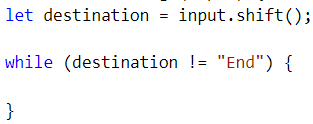
|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Вход** | **Изход** |
| Greece  1000  200  200  300  100  150  240  Spain  1200  300  500  193  423  End | Going to Greece!  Going to Spain! | France  2000  300  300  200  400  190  258  360  Portugal  1450  400  400  200  300  300  Egypt  1900  1000  280  300  500  End | Going to France!  Going to Portugal!  Going to Egypt! |

### Насоки:

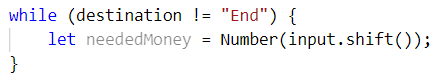
1. Вземете първата дестинация от аргумента, която Ани иска да посети:



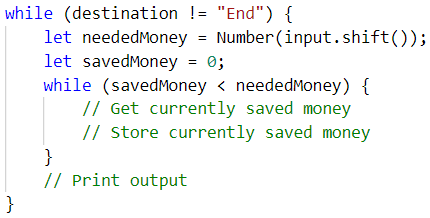
1. В while-цикъл, докато дестинацията (destination) е различна от текста “End”, извършвайте повторения:



1. За всяка дестинация извадете нужната сума пари за пътуването:



1. Създайте променлива savedMoney, в която да пазите размера на спестените пари на Ани. Във вложен while цикъл взимайте броя спестени пари и ги добявяйте към savedMoney, докато не са достатъчно, за да посети дестинацията:



## Война на имена

Напишете програма, която **изчислява ASCII стойността на няколко имена**, като **името с най-голяма стойност е победител**. Стойността на името се изчислява като съберем **ASCII стойностите на всички букви, от които се състои то**. От конзолата ще се четат имена до получаването на команда „**STOP**“, след което трябва да се изпише:

"Winner is {**името на победителя**} – {**стойността на името му**}!".

### Примерен вход и изход

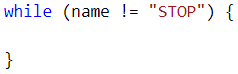
|  |  |  |
| --- | --- | --- |
| **Вход** | **Изход** | **Обяснения** |
| Petar  Georgi  Stanimir  STOP | Winner is Stanimir - 839! | Първата буква е P и тя отговаря на 80 в ASCII,  e отговаря на 101,  t отговаря на 116,  а отговаря на 97,  r отговаря на 114.  Сумата им е 508.  Продължаваме да правим същите изчисления и за останалите и полуваме, че името на Stanimir има най-голяма стойност – 839. |
| Ivo  Niki  Valio  Konstantin  STOP | Winner is Konstantin - 1065! |  |

### Насоки:

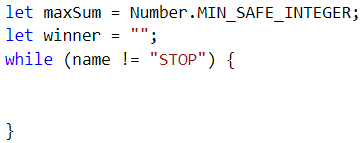
1. Вземете първото име от аргумента:



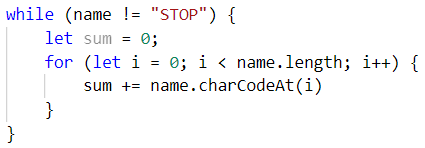
1. В while-цикъл, докато стойността на променливата name e различна от “STOP”, извършвайте повторения:



1. Създайте променлива maxSum, в която да пазите максималната сума за буквите от всяко въведено име, както и oще една променлива, в която да пазите името с най-голяма сума на ASCII стойностите на буквите:

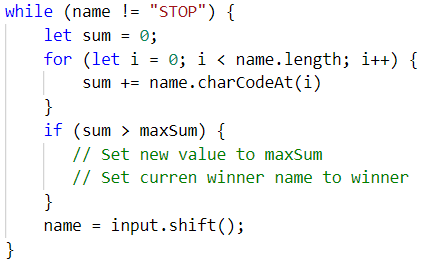


1. Направете вложен for-цикъл с крайна стойност на контролната променлива дължината на въведеното име, като достъпите неговата дължина чрез .length. Използвайте метода **charCodeAt() ,** който приема индекс, като параметър и като резултат връща цифровата репрезентация на буквата, която се намира на сътветния индекс.



Прочетете повече за структурата на string типа данни и методите, които може да използвате за достъпването на отделните символи [тук](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/charCodeAt).

1. След като **обходите всички букви в прочетеното име**, направете проверка, **дали получената сума е по-голяма от максималната** и ако е, запеметете **новата максимална** **сума** и **името** **на** **победителя** в променливата winner:



Вземето ново име от аргумента.

1. След като сте прочели всички имена и сте намерили **най-голямата сума** и **победителя**, принтирайте изхода според условието на задачата.

## Фабрика за бисквити

Фабрика за бисквити приема поръчки всеки ден. Напишете програма, която помага на сладкарите да направят бисквитите по-бързо, само като въвеждат необходимите продукти в компютъра.

Основните компоненти са **брашно, яйца и захар** и те **винаги трябва да присъстват** в сместа. За различните видове бисквити се прибавят различни допълнителни продукти**, техния брой не е ограничен**. Като **вход програмата първо ще приема** **едно цяло число - броят на партидите, които трябва да се направят днес**. На следващите редове ще се въвеждат продуктите за всяка смес.

**При въвеждане на команда "**Bake!**",** съответната смес ще се слага във фурната.

Ако сместа за печене **не съдържа поне един от задължителните компоненти : брашно, яйца или захар**, то трябва да се изписва :

"The batter should contain flour, eggs and sugar!"

, а **ако съдържа всички компоненти**:

"Baking batch number **{номера на партидата, която печем}** …".

### Примерен вход и изход

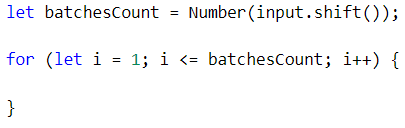
|  |  |  |  |
| --- | --- | --- | --- |
| **Вход** | **Изход** | **Вход** | **Изход** |
| 2  flour  eggs  sugar  chocolate  Bake!  flour  eggs  sugar  caramel  peanuts  Bake! | Baking batch number 1...  Baking batch number 2... | 3  flour  eggs  jam  Bake!  sugar  Bake!  flour  eggs  milk  almonds  sugar  Bake!  flour  eggs  sugar  Bake! | The batter should contain flour, eggs and sugar!  Baking batch number 1...  Baking batch number 2...  Baking batch number 3... |

### Насоки:

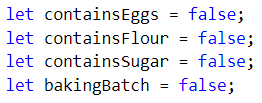
1. Вземете броя на партидите за печене –число:



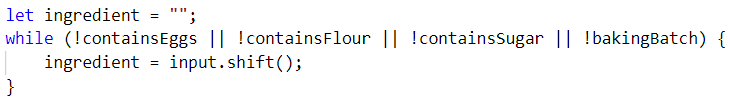
1. Създайте for-цикъл, с който да итерирате партидите:



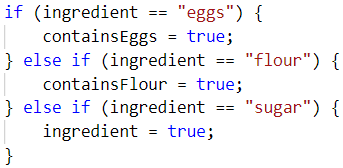
1. Създайте 4 променливи, в които да пазите резултата, дали партидата съдържа яйца, брашно захар и дали трябва да изпечете партидата, спрямо получените входни данни:



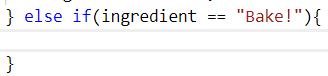
1. Създайте променлива, в която да пазите стойността на получената съставка и в while-цикъл, докато не получите **яйца**, **брашно и** **захар** за съставки на сместа, както и командата **печене** вземайте нова стойност за съставка:



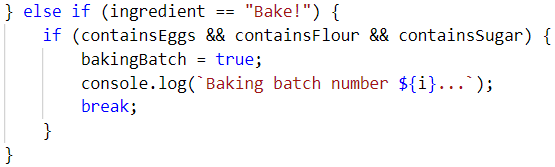
1. Направете серия от проверки за стойността на съставката. Спрямо резултата променете стойностите на променливите containsEggs, containsFlour, containsSugar:



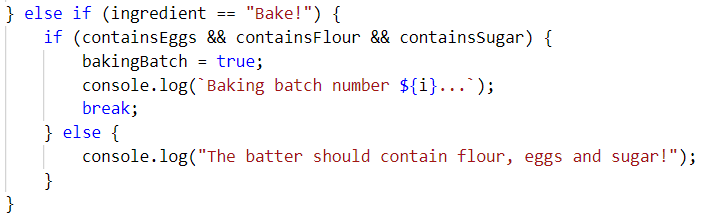
1. Добавете проверка за стойността “Bake!”:



* 1. **Ако са получени нужните съставки за печене**, печатайте изхода според условието, промените стойността на **bakingBatch** и **продължете към следващата партида**:



* 1. **Ако тестото не съдържа всички нужни съставки**, печатайте изхода според условието и **продължете към четене на още съставки**:



# Примерни изпитни задачи

## Магически числа

**Тествайте** решението си [тук.](https://judge.softuni.bg/Contests/Compete/Index/1016#8)

Да се напише програма, която чете едно цяло **„магическо“** число, въведено от потребителя, и изкарва **всички** възможни **6-цифрени числа**, за които **произведението от всички цифри** е **равно** на **„магическото“ число.**

**Пример:** „Магическо число“ -> 2

* 111112 -> 1 \* 1 \* 1 \* 1 \* 1 \* 2 = 2
* 111121 -> 1 \* 1 \* 1 \* 1 \* 2 \* 1 = 2
* 111211 -> 1 \* 1 \* 1 \* 2 \* 1 \* 1 = 2
* 112111 -> 1 \* 1 \* 2 \* 1 \* 1 \* 1 = 2
* 121111 -> 1 \* 2 \* 1 \* 1 \* 1 \* 1 = 2
* 211111 -> 2 \* 1 \* 1 \* 1 \* 1 \* 1 = 2

### Вход

Входът се чете от конзолата и се състои от **едно** **цяло число** в интервала [**1**…**600000**].

### Изход

На конзолата трябва да се отпечатат **всички “магически” числа**, разделени с **интервал**.

### Примерен вход и изход

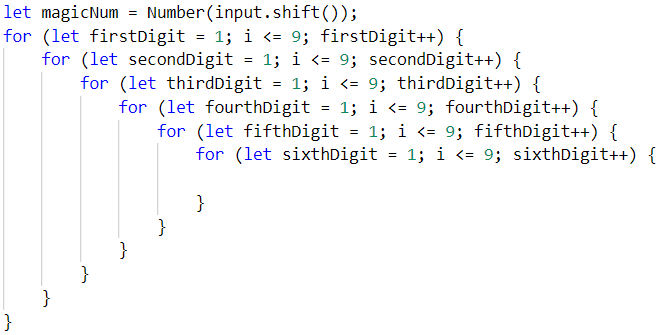
|  |  |
| --- | --- |
| **вход** | **изход** |
| 2 | 111112 111121 111211 112111 121111 211111 |
| 8 | 111118 111124 111142 111181 111214 111222 111241 111412 111421 111811 112114 112122 112141 112212 112221 112411 114112 114121 114211 118111 121114 121122 121141 121212 121221 121411 122112 122121 122211 124111 141112 141121 141211 142111 181111 211114 211122 211141 211212 211221 211411 212112 212121 212211 214111 221112 221121 221211 222111 241111 411112 411121 411211 412111 421111 811111 |
| 531441 | 999999 |

### Насоки:

1. Вземете магическото число от аргумента на функцията:



1. Създайте 6 вложени for-цикъла, които да итерират всяка възможна цифра **от 1 до 9** и за да получите комбинации от тях под формата на 6-цифрени числа:



1. Проверете, дали генерираната комбинация от цифри отговаря на условието на задачата и ако е валидна отпечатайте цифрите:

