CODE ACADEMY

ОБУЧЕНИЕ ПО ПРОГРАМИРАНЕ

. **Управление на последователността на изпълнение,** СРЕЩА 4

CODE ACADEMY

Да прегледаме как се справихте със задачите за самоподготовка.

CODE ACADEMY

Оператори, операнди, изрази

- **Оператори** – това са символи, които приемат един или повече операнди или изрази и изпълняват аритметични и логически изчисления.
- **Операндите са променливи или изрази,** които се използват в съчетание с операторите за да изчислят израз.
- Комбинация отоперанди и оператори формира **израз**.
- Изразите са последователност от оператори, операнди и пунктуации, определящи изчисленията.
- Изчислението на изрази се базира на оперторите, които съдържа израза и контекста в който се използват.
- Изразите могат да резултират в стойност или да произведат **странични ефекти**.
- **страничен ефект е** промяна в състоянието на средата за изпълнение..

CODE ACADEMY

Оператори, операнди, изрази

Съставен оператор на присвояване	Пример	Еквивалентен израз
+=	nindex += 3	index = nindex + 3
-=	*(paPter++) -= 1	* paPter = *(paPter ++) - 1
*=	fbonus *= fpercent	fbonus = fbonus * fpercent
/=	ftimePeriod /= fhours	ftimePeriod = ftimePeriod / fhours
%=	fallowance %= 80	fallowance = fallowance % 80
<<=	iresult <<= inum	iresult = iresult << inum
>>=	byleftForm >>= 1	byleftForm = byleftForm >> 1
&=	bybitMask &= 2	bybitMask = bybitMask & 2
^=	itestSet ^= imainTest	itestSet = itestSet ^ imainTest
|=	bflag |= bonBit	bflag = bflag | bonBit

CODE ACADEMY

Приоритет на операторите

Symbol	Type of Operation	Associativity
`[ ] ( ) . ->` postfix `++` and postfix `--`	Expression	Left to right
prefix `++` and prefix `--` `sizeof` `&  *  + - ~ !`	Unary	Right to left
typecasts	Unary	Right to left
`* / %`	Multiplicative	Left to right
`+ -`	Additive	Left to right
`<< >>`	Bitwise shift	Left to right
`< > <= >=`	Relational	Left to right
`== !=`	Equality	Left to right
`&`	Bitwise-AND	Left to right
`^`	Bitwise-exclusive-OR	Left to right
`|`	Bitwise-inclusive-OR	Left to right
`&&`	Logical-AND	Left to right
`||`	Logical-OR	Left to right
`? :`	Conditional-expression	Right to left

`= *= /= %=`
`+=  -=  <<=  >>= &=`
`^= |=`
	Simple and compound assignment	Right to left

Аритметични оператори

operator	значение	примери
+	събиране	x=3+2; /∗constants∗/ y+z; /∗variables∗/ x+y+2; /∗both∗/
-	изваждане	3−2; /∗constants∗/ **int** x=y−z; /∗variables∗/ y−2−z; /∗both∗/
*	умножение	**int** x=3∗2; /∗constants∗/ **int** x=y∗z; /∗variables∗/ x∗y∗2; /∗both∗/
/	деление	**float** x=3/2; /∗produces x=1 (int /) ∗/ **float** x=3.0/2 /∗produces x=1.5 (float /) ∗/ **int** x=3.0/2; /∗produces x=1 (int conversion)∗/

CODE ACADEMY

Оператори за сравнение

Операторите за сравнение сравняват два операнда и произвеждат булев или логически резултат. В езика С всяка ненулева стойност (1 по конвенция) се счита за ’true’ , докато 0 се разглежда като false.

оператор	значение	примери
>	по-голямо от	3>2; /∗дава1 ∗/ 2.99>3 /∗дава 0 ∗/
>=	по-голямо или равно от	3>=3; /∗дава 1 ∗/ 2.99>=3 /∗дава 0 ∗/
<	по-малко от	3<3; /∗дава 0 ∗/ ’A’<’B’ /∗дава 1∗/
<=	по-малко или равно от	3<=3; /∗дава 1 ∗/ 3.99<3 /∗дава 0 ∗/

Проверка за равенство

Проверката за равенство е един от най-използваните оператори за сравнение

Оператор	значение	примери
==	равно ли е	3==3; /* дава 1 */ 'A'=='a' /*дава 0 */
!=	различно ли е	3!=3; /*дава 0 */ 2.99!=3 /*дава 1 */

Оператора за сравнение == е различен от оператора за присвояване =

! За реални числа от тип float == не се прилага, т.к. е с ограничена точност

CODE ACADEMY

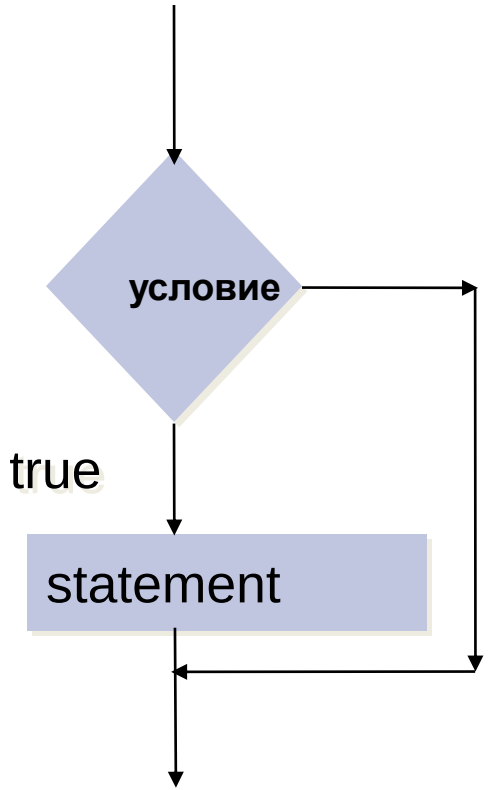
Логически оператори

Оператор	значение	примери
&&	AND	((9/3)==3) && (2*3==6); /* дава 1 */ ('A'=='a') && (3==3) /*дава 0 */
||	OR	2.99>=3 || 0 /*дава 0 */ 2==3 || 'A'=='A'; /*дава 1 */
!	NOT	!(3==3); /*дава 0 */ / !(2.99>=3) /*дава 1 */

Изчислението на израза се прекратява, ако стойността на условния израз може да се определи по-рано.

Условен оператор

```
if (condition)
        statement;
```



Управление на изпълнението

● Изразът if – else се използва за изразяване на решения. Официалният синтаксис е

if (<условие>) {
/* изпълнява се при изпълнено условие */
оператор1
} else {
/* изпълнява се при неизпълнено условие */
оператор2.
}

● като частта else не е задължителна. Изразът се изчислява и, ако той е истина, т.е. има не нулева стойност, се изпълнява оператор 1. Ако не е истина т.е. е нула, се осъществява else и се изпълнява оператор 2. Препоръчва се да се използват блокове { }.

Управление на изпълнението

Когато има влагане н условен оператор, ако искате else да се свързва с първия if трябва да използвате фигурни скоби{ }, за да подскажете правилната асоциация. Например така:

```
if (n > 0){

        if (a > b)
                z = a;
}
    else{
        z = b;
    }
```

Условна конструкция if-else-if

```
if ( <условие-1> ) {
/* изпълнява се при изпълнено условие-1
} else if ( <условие-2> ) {
/* изпълнява се при изпълнено условие-2
} else if ( <условие-3> ) {
/* изпълнява се при изпълнено условие-3
} else {
/* изпълнява се при неизпълнени условия
}
```

Тази последователност от if-ове е най-често срещаният начин да напишете решение с множество варианти. Изразите се изчисляват подред: ако някой от тях е истина се изпълнява това което е в него и с това се прекратява цялата верига.

CODE ACADEMY

Задачи:

1. Използвайте конструкцията if-else if – else за да принтирате един от 3 възможни отговора.

Press 1 to see message Hello
Press 2 to see Poem
Press 3 to see hidden mesage

В началото, с помощта на функцията printf(), попитайте ползвателя на вашата програма кое съобщение иска да види, като натисне 1, 2 или 3 . След това с помощта на функцията scanf(), запишете неговия избор, след което използвайте if-else if – else конструкцията, за да принтирате отговора.

CODE ACADEMY

Задачи:

2. Да се напише програма, която чете едно произволно цяло число, отделя последната му цифра и проверява дали тази цифра е 0 или 5, т.е. дали това число се дели на 5. В случай, че числото се дели на 5, се извежда съобщението “Deli se na 5”, в противен случай извежда съобщението “Ne se deli na 5”.

3. Напишете C програма за въвеждане на символ от потребителя и проверете дали знакът е с главни или малки букви, като използвате if else. Проверката за главни и малки букви ще стане като се използват кодовата таблица за символите и факта, че кодовете на последователните символи са последователни.

CODE ACADEMY

Задачи:

4. Да се напише програма, която да въведе номера на месеца и да изведе броят дни в този месец.

5. Напишете C програма, която въвежда оценките по 5 предмета, като спазва следните условия
If percentage >= 90% : Grade A
If percentage >= 80% : Grade B
If percentage >= 70% : Grade C
If percentage >= 60% : Grade D
If percentage >= 40% : Grade E
If percentage < 40% : Grade F

пример
Input
```
Input marks of five subjects: 95 95 97 98 90
```
Output
```
Percentage = 95.00 Grade A
```

CODE ACADEMY

Конструкция switch

Операторът switch е начинът за изразяване на решение с много варианти, който проверява дали даден израз съответства на някоя от константите (цели стойности) и като намери съответствие, преминава през този клон на изпълнение.

Конструкция switch

```
switch (<променлива>) {
case <стойност-1> : {
        /* <променлива> == <стойност-1> */
        break;
}
case <стойност-2> : {
        /* <променлива> == <стойност-2> */
        break;
}
default : {
        /* <променлива> != <стойност-1/2>
        break;
}
}
```

CODE ACADEMY

Задачи:

6. Напишете програма на C за въвеждане на номер на ден от седмицата (1-7) и отпечатване на името на деня от седмицата с помощта на главни букви.

7. Напишете C програма, която въввежда оценките по 5 предмета, като спазва следните условия
If percentage >= 90% : Grade A
If percentage >= 80% : Grade B
If percentage >= 70% : Grade C
If percentage >= 60% : Grade D
If percentage >= 40% : Grade E
If percentage < 40% : Grade F

пример
Input
```
Input marks of five subjects: 95 95 97 98 90
```
Output
```
Percentage = 95.00 Grade A
```

CODE ACADEMY

Конструкция switch

Всеки случай – **case**(а), съдържа една или повече целочислени константи или константни изрази. Ако случаят съответства на стойността в израза, изпълнението започва от този случай. Всички изрази обозначаващи случаите, трябва да са различни. Случаят обозначен с **default** се изпълнява, когато нито един от случаите не е удовлетворен. Не е задължително да има default случай, ако той не съществува и същевременно нито един от горните случаи не е удовлетворен, не се случва нищо.

CODE ACADEMY

Задачи:

8. Напишете програма на C за въвеждане на номер на седмицата (1-7) и отпечатване на името на деня от седмицата с помощта на главни букви

Input

```
Inputweek number(1-7): 2
```

Output

```
Tuesday
```

9 Напишете програма на C, за да създадете калкулатор, който изпълнява основни аритметични операции (събиране, изваждане, умножение и деление). Калкулаторът трябва да въведе две числа и оператор от потребителя. Той трябва да извършва операция според въведения оператор и трябва да приема вход в даден формат.

<номер 1> <оператор> <номер 2>

Input

```
5.2 - 3
```

Output

```
2.2
```

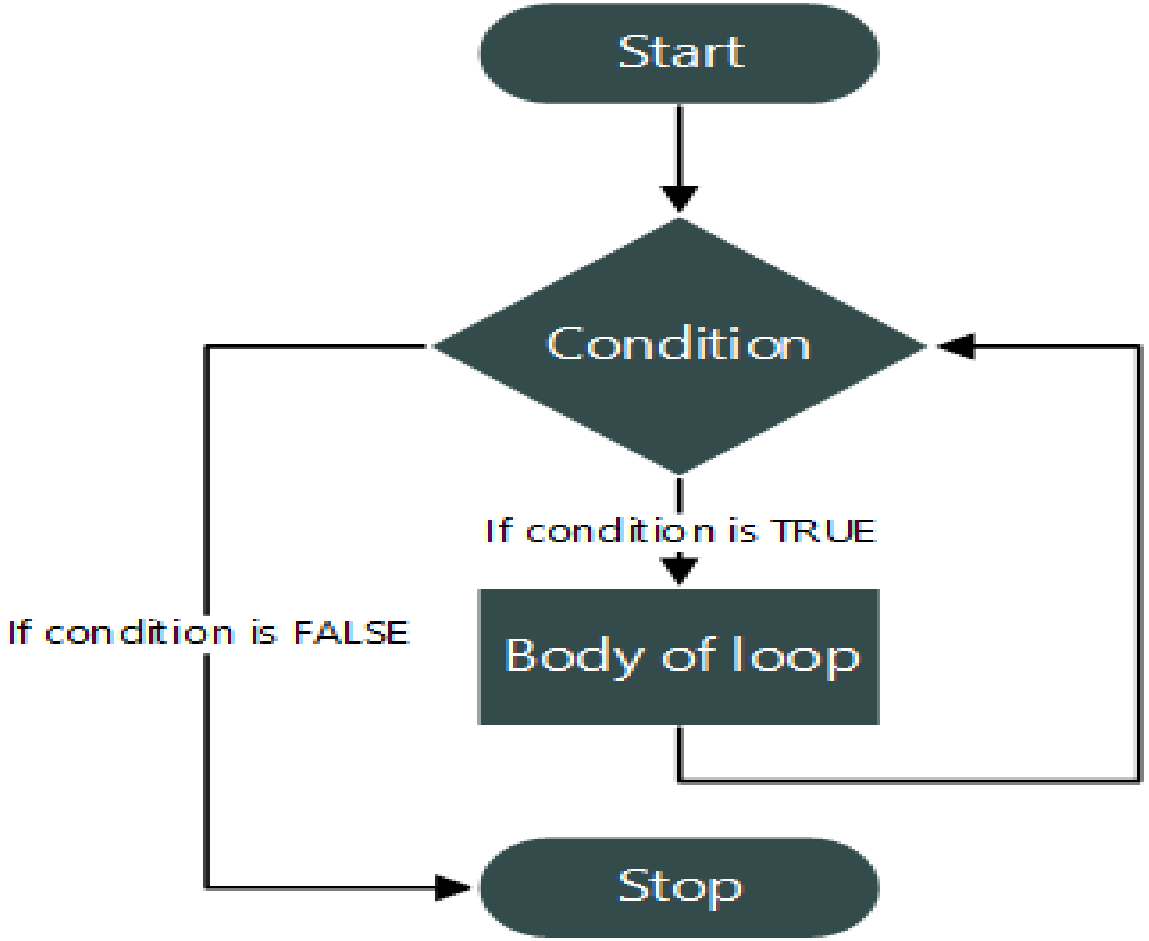
Цикъл while

```
while (<условие>) {
/* изпълнява ако <условие> е изпълнено */
}
```

- Цикълът приключва, когато условието не е изпълнено.
- Условието се проверява преди да се изпълни тялото на цикъла.

CODE ACADEMY

Цикъл while



CODE ACADEMY

Задачи с цикъл while

Задача 10 Напишете програма на C за отпечатване на естествени числа от 1 до 10 с помощта на цикъл while.

Задача 11 Напишете програма на C за отпечатване на азбуката от a до z с помощта на цикъл while.

Задача 12 Напишете програма на C, която с помощта на цикъл while намира gcd(a,b) – най-големият общ делител на две числа **a** и **b**, които се въвеждат.
За целта използвайте алгоритъма на Паскал
Ако двете числа са равни, то всяко от тях е НОД, иначе от по-голямото изваждаме по-малкото, докато двете станат равни.

Задача 13 Да се изчисли сумата на целите числа в зададен диапазон [a, b] – използване на цикъл while.

Цикъл do - while

```
do {
/* изпълнява се тялото */
} while (<условие>);
```

- Тялото на цикъла ще бъде изпълнено поне веднъж.

- Цикълът продължава, докато условието е изпълнено.

Цикъл do - while

Задача 13 Да се изчисли сумата на целите числа в зададен диапазон [a, b] – използване на цикъл **do-while**

```
#include <stdio.h>
int main () {
int nValue = 13;
do {
      printf("Value: %d\n", nValue);
      nValue ++;
} while( nValue < 20 );
return 0;
}
```

Задачи с цикъл do while

Задача 14 Напишете програма на С, която прочита едно цяло число и като използва цикъл do while, намира и извежда броят на цифрите му.

Задача 15 Как да проверите дали една дума дали е палиндром?

Трябва да четем отляво и отдясно, ако думата е една и съща, независимо дали я четем отляво или отдясно, тогава се нарича палиндром.

Пример за палиндром „боб“, 12321 .

Напишете програма на С , която да покаже дали едно 5 цифрено число е палиндром или не, използвайки цикъл do-while?

CODE ACADEMY

Оператор за цикъл for(; ;)

Операторът за цикъл **for** има управляваща променлива или променлива брояч, която брои колко пъти сме минали през цикъла, т.е. колко пъти сме повторили тази част от кода.

for (int i = начална стойност ; условие за повтаряне ; i = i + step){ код }

Тук, управляващата променлива е i. **Областта на действие** на тази променлива е само в границите на цикъла for. В момента, когато се излезе от цикъла, променливата не съществува. Това означава, че извън цикъла нейната стойност не може да се използва.

Началната стойност на i е тази стойност, при която кодът от цикъла се изпълнява за първи път. След това, за да се повтори този код, се проверява дали е изпълнено условието на цикъла.

Условието в цикъла for е условие за повтаряне. Ако това условие е изпълнено, стойността на променливата i се увеличава със стойността на променливата step. След това кодът от цикъла отново се повтаря и така нататък, докато при текущата стойност на променливата i условието на цикъла не е изпълнено.

Ако условието на цикъла е нарушено, тогава се излиза от цикъла и се изпълнява първият оператор, който е написан непосредствено след **оператора for**.

CODE ACADEMY

Цикъл for (; ;)

for (<инициализация> ; <условие> ; <стъпка>) {
/* изпълнява се, ако <условие> е изпълнено */
}

Стъпката, с която се променя управляващата променлива може да бъде както нагоре, така и надолу.

Може да имаме i = i − 2, но това е обвързано както с началната стойност, така и с условието.

Трябва много да се внимава, за **да не се получи невъзможен или безкраен цикъл**.

Когато променливата i се увеличава с 1, вместо i = i + 1 често пишем i + +, а когато променливата i се намалява с 1 вместо i = i−1, пишем i− −.

Има и други такива съкръщения - вместо s = s+4, пишем s+ = 4, вместо s = s − 4, пишем s− = 4.

Името на управляващата променлива може да бъде произволно.
Често се използва променлива с име i - идва от думата итерация(iteration) - повторение.

CODE ACADEMY

Задачи с цикъл for

Задача 16. Принтирайте числата от 13 до 19 като използвате **for цикъл.**

Задача 17. Да се напише програма, която въвежда едно цяло положително число n и намира и извежда сумата на всички числа от 1 до n**.**

Задача 18. Да се напише програма, която намира сумата на всички числа, които се делят на 7 и са в интервала между числото a и числото b. Целите числа a и b се задават на вход

Задача 19. За да стане голям един заек му е необходим един месец. След още един месец се ражда още един заек и така зайците стават 2. След още един месец първият заек ражда още едно ново зайче, а вторият е пораснал и така зайците са станали 3. След още един месец първите два заека раждат по още един нов заек, а третият порасва и така нататък.
По този начин се получава известната редица на Фибоначи от числата 1, 1, 2, 3, 5, 8, 13, . . .
Характерното за тази редица е, че всяко следващо число от нея се получава като се сумират предходните две числа. Да се напише програма, която чете едно цяло положително число n и извежда n-тото число от тази редица

Задачи с цикъл for

Задача 20. Да се напише програма, която намира произведението на всички нечетни числа, които са по-малки от n. Числото n се задава на вход.

Задача 21. Да се напише програма, която реализира играта Бикове и Крави

1234

2156 → 2 крави

1256 → 2 бика

1247 → 2 бика 1 крава

Пропускане на итерация

- За пропускане на итерация на цикъл се използва ключовата дума continue.
- При вложение цикли се прекъсва само текущият.

Пример:

```
int iValue = 0;
while(iValue < 13) {
iValue ++;
if (iValue == 5)
        continue;
    //some code
}
```

Прекъсване на цикъл

- За прекъсване на цикъл се използва ключовата дума break.
- При вложение цикли се прекъсва само текущия.

Пример:

```
int iValue = 0;
for( ; ; ) {
iValue ++;
if (iValue > 5)
        break;
}
```

CODE ACADEMY

Прекъсване на цикъл

Задача 22. Създайте безкраен цикъл с for (;;) и принтирайте числото, което на всяка итерация на цикъла се увеличава с едно. Когато числото стигне 48, излезте от цикъла с break.

CODE ACADEMY

Безусловен преход goto

- Преходът се изпълнява в рамките на блок.
- Използването на goto влошава четимостта на програмата

```
goto labelIdentifier;
/* C statements */
labelIdentifier: C-statement;
```

CODE ACADEMY

Символен вход и изход

Моделът за вход и изход се поддържа от стандартната библиотека

#include <stdio.h>

Текстовият вход и изход се разглежда като поток от символи – последователност от символи, разделени в редове.

<stdio.h> предоставя няколко функции за четене и писане на един символ.

getchar() - всеки път, когато бъде извикана, чете следващия символ от текстовия поток (обикновено символите идват от клавиатурата)

putchar() - отпечатва символа на екрана**.**

CODE ACADEMY

Символен вход и изход

Без да знаете нищо друго за входа и изхода, освен функциите getchar и putchar, можете да напишете удивително голямо количество полезен код.

Пример

Четене и писане на символ по символ чрез променливата char c;

входът, записан в “c”, се печати на изхода.

```
#include <stdio.h>

int main(void){
char c;
c = getchar();
while(c != EOF){
        putchar(c);
        c = getchar();
}
}
```

CODE ACADEMY

Символен вход и изход

Релационният оператор != означава различно

while(c != EOF)

проверява дали символът, който пишем от клавиатурата, не е символ за край на файл.

Край на файл - подаван от клавиатурата е

Ctrl + d → за Lunux

Ctrl + c → за Windows.

Символите на клавиатурата или на екрана → в паметта е комбинация от битове.

```
printf("Character literals: '%c'\n", 65); ---→  'A'
```

Типът char е предназначен за съхранение на символни данни и има диапазон от 0 до 255.

Но символите са значително повече.

CODE ACADEMY

Символен вход и изход

Вместо типа char може да се използва и всеки друг целочислен тип,
   който е достатъчно голям,за да побере символите по-големи от 255.

За да може да се принтира и кирилица,
     използваме тип int за променлива за символи с;

CODE ACADEMY

Преброяване на символи

Когато искаме да принтираме стойността на символа **c** използваме **putchar(c).**

Докато пишем в конзолата, стойността на променливата **nc** се увеличава с **1.**

Когато прекъснем цикъла с **Ctrl + c** или със **Ctrl + d** получаваме резултата.

nc++; може да се замени с **nc = nc + 1;** или с **++nc;**

CODE ACADEMY

Преброяване на редове

Стандартната библиотека **<stdio.h>**
осигурява входния текстов поток да се появява като последователност от редове.

Всеки ред завършва със символ за нов ред - '\n' .

Следователно,
редовете могат да се преброят като се преброят символите за нов ред - '\n' .

Когато се пише в конзолата, всяко натискане на нов ред ни дава символа '\n' .

Така, ако в тялото на while поставим един if , който проверява дали символът е '\n'. Оператора == и ако c в c == '\n' се инкрементира променливата nl, ако c не е равно на '\n' , не се влиза в тялото на if конструкцията - if(c == '\n') и променливата nl не се увеличава с 1.

CODE ACADEMY

Приоритет на оператори

Приоритета на операторите реално се ползва по следния начин в условната конструкция. Нека се върнем към първия израз и да видим как може да го опростим и напишем на по малко редове, използвайки приоритета на операциите:

```
#include <stdio.h>

int main(void){
int c = 0;
while((c = getchar()) != EOF)
       putchar(c);
return 0;
}
```

Така нашият код се съкрати до 8 реда.

CODE ACADEMY

Приоритет на оператори

В израза a = c = 5 на a се присвоява резултатът от c = 5.

Изразът c = getchar(); има определена стойност, която остава в стойността на c - отляво на знака за присвояване.

==> Присвояването може да се използва и като част от по- голям израз.

Затова и присвояването на символ на променливата **c,** може да се постави в условната част на цикъла **while** без проблем.

Операторът за присвояване, когато има операции при, които се изчисляват и изрази **има по малък приоритет от оператора за сревнение !=** .

Ако сложим присвояването без ограждащи скоби **while(c = getchar() != EOF)**
първо ще се извърши сравнението getchar() != EOF и **след това ще се присвои резултатът**, който връща този оператор true(1) или false(0).
Или вместо стойността на някакъв символ (примерно 'A'(65)), на c ще се присвои 1 или 0.

Затова слагаме скоби, за да извършим първо оператора за присвояване и след това оператора за сравнение - while((c = getchar()) != EOF)

CODE ACADEMY

Задача 1

Пребройте символите, подавани на конзолата, с функцията getchar();

Преброяване на символи

```
#include <stdio.h>

int main(){
char c;
int count=0;
while((c = getchar()) != EOF){
     count++;
}
printf("%d\n",count);

}
```

CODE ACADEMY

Задача 2

Пребройте редовете, подадени чрез текст на конзолата. Използвайте функцията getchar();

CODE ACADEMY

Преброяване на редове

```
#include <stdio.h>

int main(){
char c;
int countRow=0;
while((c = getchar()) != EOF){
       if(c=='\n')
     countRow++;
}
printf("%d\n",countRow);

}
```

CODE ACADEMY

Задачи:

3.Напишете програма, която преброява шпациите, табулациите и новите редове.

4. Напишете програма, която копира входа си на изхода, като замества всеки низ повече от една шпации, с една единствена шпация.

CODE ACADEMY

Преброяване на табулации и спейсове

```
#include <stdio.h>

int main(){
char c;
int countTab=0,countSpace=0;
while((c = getchar()) != EOF){
       if(c=='\t')
            countTab++;
        if(c==' ')
            countSpace++;
}
printf("%d\t%d\n",countTab,countSpace);


}
```

CODE ACADEMY

Оставяне на единствен спейс

```
#include <stdio.h>

int main(){
char c;
int countTab=0,countSpace=0;
while((c = getchar()) != EOF){
       if(c=='\t')
            countTab++;
        if(c==' ')
            countSpace++;
}
printf("%d\t%d\n",countTab,countSpace);


}
```

CODE ACADEMY

Задачи за самоподготовка

1. Клиент вложил A лв с месечна лихва B%. Как ще се променя тази вноска, ако клиентът не тегли
пари през този период:
а/за 5 години;
б/за N години;
в/докато вноската стане по-голяма от C лв.
Да се напише програма. Числата са на вход.

2. Започвайки тренировка спортист пробягал първия ден A км. Всеки следващ ден той увеличавал дневната си норма с B% спрямо предишния ден. Какъв сумарен път ще пробяга спортистът за :
а/10 дни;
б/N дни;
в/докато пробягания за ден път стане по-голям от X км и на кой ден;
г/докато общо пробягания път стане по-голям от У км и за колко дни.
Да се напише програма. Числата са на вход.

CODE ACADEMY

Задачи за самоподготовка

3. По зададено естествено число N, да се намерят всички двойки естествени числа М и К, за които
М2+К2=N.

4. Дадени са a, b, N (b !=0, 1<=N<=100). Да се намери частното a/b и да се отпечата на екрана с N цифри след десетичната запетая.

5. Да се намери сумата от тези елементи на редица, които са удвоени нечетни числа, ако:
а/редицата е с 41 елемента;
б/първо се въвежда броят на елементите на редицата;
в/редицата е последен елемент 0;
г/редицата е с елементи, чиято сума е по-голяма от 999.

CODE ACADEMY

Задачи за самоподготовка

6. Да се изведат първите N числа на една редица (аритметична прогресия), ако са дадени първият член на редицата и разликата между първите два елемента. Да се напише програма. Числата са на вход.

7. Да се изведат първите N числа на една редица (геометрична прогресия), ако са дадени първият член на редицата и частното между първите два елемента. Да се напише програма. Числата са на вход.

8. По дадено N да се изчисли A(0)+A(1)+A(2)+...+A(N), ако A(0)=1,A(K+1)=2.A(K)+1 за K>=1.

Да се изчисли с точност 6 знака след запетаята числото П по формула на Готфрид Лайбниц от 1673г:

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} + - \ldots$$