



ФАКУЛТЕТ ЗА ЕЛЕКТРОТЕХНИКА И ИНФОРМАЦИСКИ ТЕХНОЛОГИИ

# Наредби за контрола на тек

# if - else

Општиот облик на if наредбата е следниот

```
if (uslov)
    naredba_za_vistinit_uslov;
else
    naredba_za_nevistinit_uslov;
```

Ако има блокови на наредби тогаш се  
означува почетокот и крајот на блокот

```
if (uslov)
{
    blok_naredbi_za_vistinit_uslov;
}
else
{
    blok_naredbi_za_nevistinit_uslov;
}
```

Делот else не мора да постои.

```
if (uslov)
    naredba_za_vistinit_uslov;
```

и

```
if (uslov)
{
    blok_naredbi_za_vistinit_uslov;
}
```

Условот во заградата може да  
биде каков било аритметичко  
логички израз.

```

if (sredstva > cena)
    kupi;
else
    if (imas_prijatel)
        pozajmi_pari;
    else
        najdi_rabota;

```

```

if (x > 5)
    if(x < 11)
        polozi;
else
    padna;

```

?

---

```

#include <stdio.h>
int main()
{
    char c;
    printf( "Vnesi bukva: ");
    scanf("%c",&c);
    if( c == 'a' || c == 'e' || c == 'i' || c == 'o'
        || c == 'u' )
        printf( "Vnesena e samoglaska\n");
    printf("\n");
}

```

```

if (x > 5){
    if(x < 11) polozi;
}
else    padna;

```

Често се пишува

```
if(izraz)
```

наместо

```
if(izraz != 0)
```

ИЛИ

```
if(!izraz) наместо if(izraz == 0)
```

Во изразите може да се најдат и наредби за доделување и/или инкрементирање/декрементирање

```
if((c=getchar()) != '\n' && ++i < n) ...
```

# Циклуси

Циклусите се употребуваат за повторување групи наредби сè додека некој услов е исполнет.

```
while (uslov)
    naredba;
```

ИЛИ

```
while (uslov)
{
    blok_naredbi;
}
```

Условот се испитува на почетокот (уште пред влезот во циклусот). Наредбите од циклусот се повторуваат ниту еднаш или повеќе пати.

Пример:

```
while (parite < cena_na_kola)
    raboti_poveke;
kupi_kola;
```

# do - while

do

```
    naredba;  
while (uslov);
```

Слична на наредбата REPEAT - UNTIL со таа разлика што наредбите од циклусот се повторуваат сè додека условот **е** исполнет.

ИЛИ

```
do  
{  
    blok_naredbi;  
}  
while(uslov);
```

Условот се испитува на крајот, поради што блокот на наредби се извршува **најмалку еднаш!**

Пример програма:

```
#include <stdio.h>
int main()
{
    int n = 1;
    int broj, suma = 0;
    while( n <= 5 )
    {
        printf( "Vnesi broj: ");
        scanf("%d", &broj);
        suma += broj;
        n++;
    }
    printf("\nSredna vrednost na vnesenite broevi"
           "\n e %f\n", (float)suma / (n-1));
    return(0);
}
```

# for циклуси

**for** циклусот во C се дефинира во три дела на следниот начин:

```
for(inicijalizacija ; uslovi ; inkrementi_ili_dekrementi)  
    naredba;
```

ИЛИ

```
for(inicijalizacija ; uslovi ; inkrementi_ili_dekrementi)  
{  
    blok_naredbi;  
}
```



Во делот за иницијализација вообичаено се доделуваат почетните вредности на бројачите.

```
for(x = 1; ...
```

Ако има повеќе бројачи нивните иницијализации се одделуваат со запирки.

```
for(x = 1, a = 0, z = start; ...
```

Условите обично се логички и релациски изрази за поставување на некои услови кои ќе го контролираат извршувањето на циклусот.

Сè додека условот е исполнет се инкрементираат или декрементираат бројачите и се повторуваат наредбите од циклусот.

```
for (j = 0; j < 30000; j++)
```

Во третиот дел обично се инкрементираат или декрементираат една или повеќе променливи, но може да се стави и која и да е друга наредба.

```
for (x = 0, j = 0; j < 100; j++, x+=5)
```

# Извршување на for наредба

- Во секој од овие делови може да се стават произволни наредби, но редоследот на нивното извршување и интерпретацијата на нивните резултати е точно одреден:
- Наредбите од делот *inicijalizacija* се извршуваат точно еднаш, на почетокот - пред влезот во циклусот;

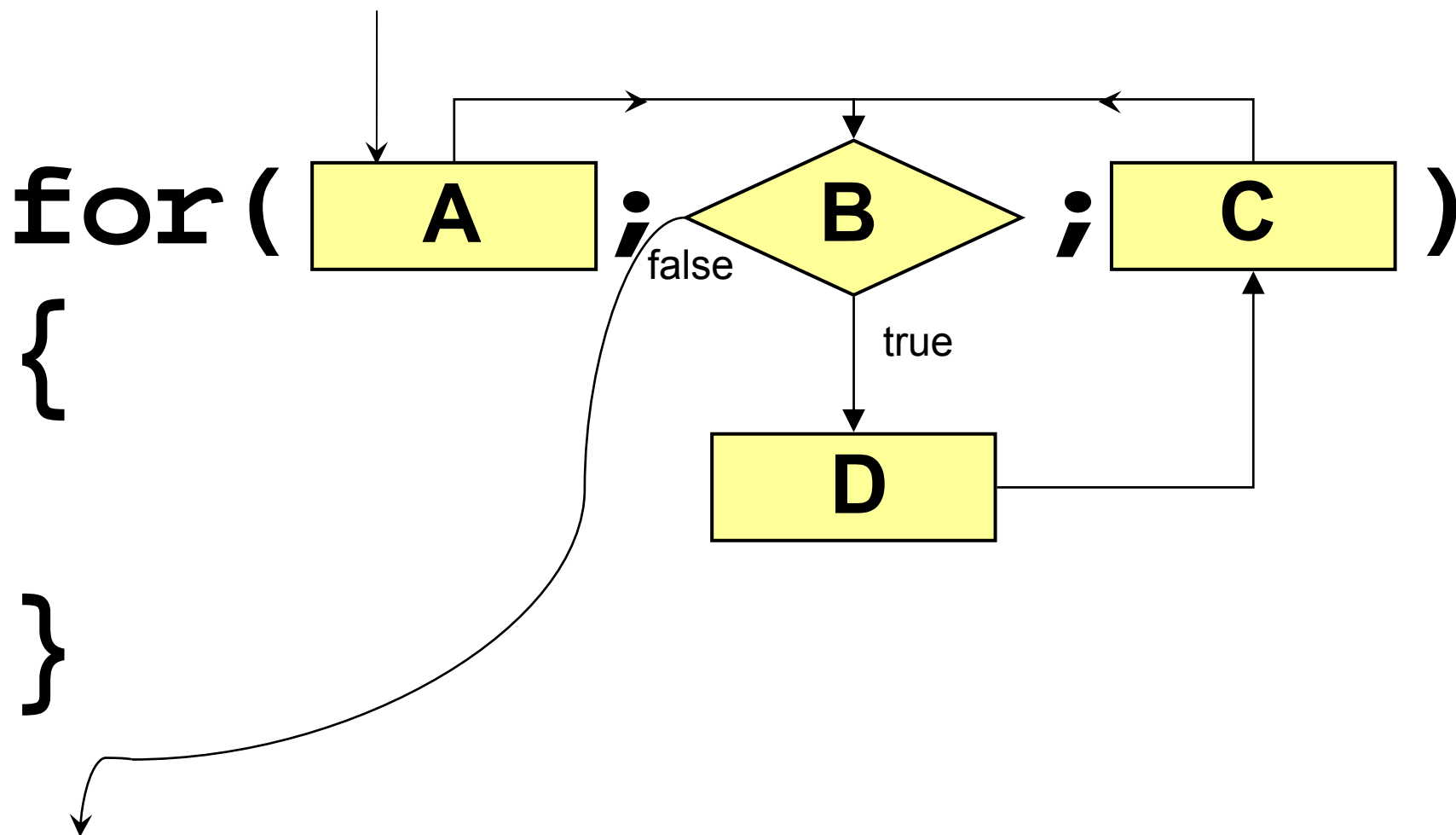
# Извршување на for наредба

- Наредбите од делот *uslovi* се извршуваат пред почетокот на секој нов циклус и ако резултираат со вредност која се интерпретира како логичка вистина се повторуваат наредбите од циклусот, инаку се завршува повторувањето на циклусот и се продолжува со наредбите зад (по) циклусот;

# Извршување на for наредба

- Наредбите од делот *inkrementi\_ili\_dekrementi* се извршуваат на крајот на секој циклус (по извршувањето на сите наредби од телото на циклусот *blok\_naredbi*) по што се извршуваат наредбите од делот *uslovi* и ако се задоволени, циклусот се повторува.

# Редослед на извршување на деловите од for наредбата



```

#include <stdio.h>
int main()
{
    int i=0;
    printf("Ke pocne ciklus...\n");
    for(printf("A");
        printf("B"),i<3;
        printf("C"),i++)
        printf("D");
    printf("\nCiklusot zavrshi.");
    return 0;
}

```

```

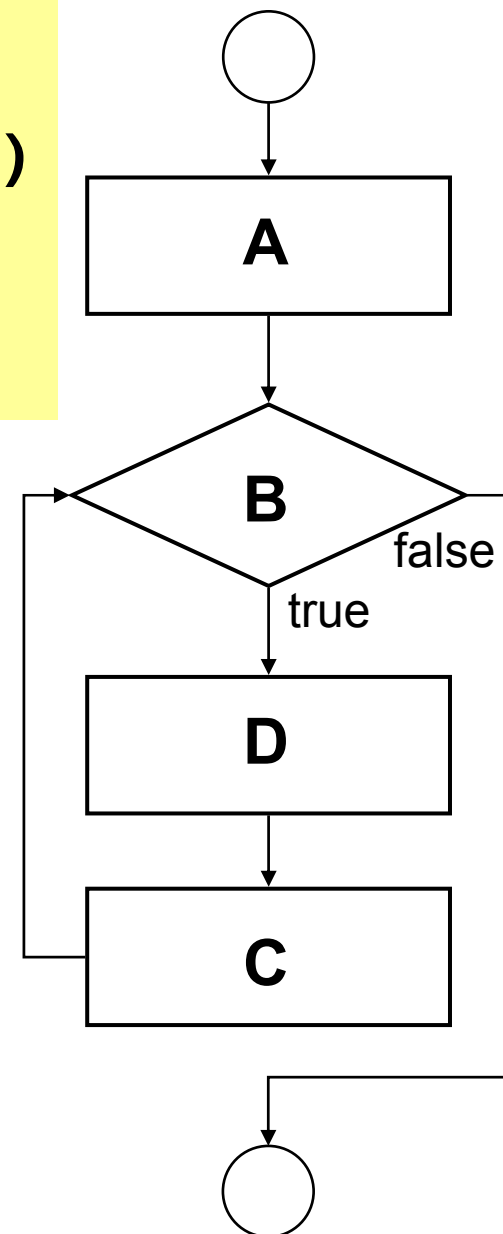
. . .
for(A ; B ; C)
    D;
. . .

```

```

Ke pocne ci kl us...
ABDCBDCBDCB
Ci kl usot zavrshi .

```



Што печати следнава програма?

```
#include <stdio.h>
int main()
{
    int j = 0;
    while (j++ < 3)
        printf( "Ha ");
    do
    {
        j -= 2;
        printf( "Hi ");
    }
    while ( ++j );
    for(j = 1; j <= 3; j++)
        printf( "Ho ");
    printf("\n");
    return 0;
}
```

j = 2

Ha Ha Ha Hi Hi Hi Hi Ho Ho Ho



Кај for наредбата треба да се внимава на тоа дека таа нема ; на крајот.

```
#include <stdio.h>
int main()
{
    int suma, x, y;
    suma = 0;
    y = 5;
    for (x = 1; x < y; x++);
        suma = suma + x * y;
    printf(" Sumata e %d", suma);
    return 0;
}
```

Sumata e 25

Некои делови на for наредбата можат да бидат празни:

```
#include <stdio.h>
int main()
{
    int c;
    printf("Vnesuvaj znakovi:\n(Vnesi x za kraj)\n");
    for (      ; c != 'x';      )
    {
        c = getchar();
        putchar(c);
    }
    printf("\nKraj na ciklusot!\n");
    return 0;
}
```

## Илустрација на употреба на влезно - излезна наредба во рамките на for наредба

```
#include <stdio.h>
int main()
{
    int broj=0;
    for(printf( "vnesuvaj broevi\n" );
        broj != 6;
        scanf( "%d", &broj ));
    printf( "Toj broj go sakam!\n" );
    return 0;
}
```

# Пример програма со повеќе броила:

```
#include <stdio.h>

int main( )
{
    int i, j;
    for (i=0, j=8; i<8; i++, j--)
        printf("%d + %d = %d\n", i, j, i+j);
    return 0;
}
```

0	+	8	=	8
1	+	7	=	8
2	+	6	=	8
3	+	5	=	8
4	+	4	=	8
5	+	3	=	8
6	+	2	=	8
7	+	1	=	8

## Програма која ги печати фибоначевите броеви помали од 1000.

```
#include <stdio.h>
int main()
{
    int c1, c2;
    const int n = 1000;
    for(printf("%d ", (c1=c2=1)); c2<n ;
        c1=(c2+=c1)-c1)
        printf("%d ", c2);
    printf("\n");
    return(0);
}
```

1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987

# Наредба за излегување од циклус - **break**

Наредбата `break` овозможува излегување од циклус реализиран со `for`, `while`, `do-while` или од наредбата `switch` пред условот за напуштање на циклусот да биде исполнет.

```
#include <stdio.h>
int main()
{
    int x = 0;
    for (;;)
    {
        if(x > 20000)
            break;
    }
    printf("%d", x);
    return(0);
}
```

# Наредба - continue

Наредбата `continue` не носи директно во следниот чекор на јамката запоставувајќи ги наредбите до крајот на јамката.

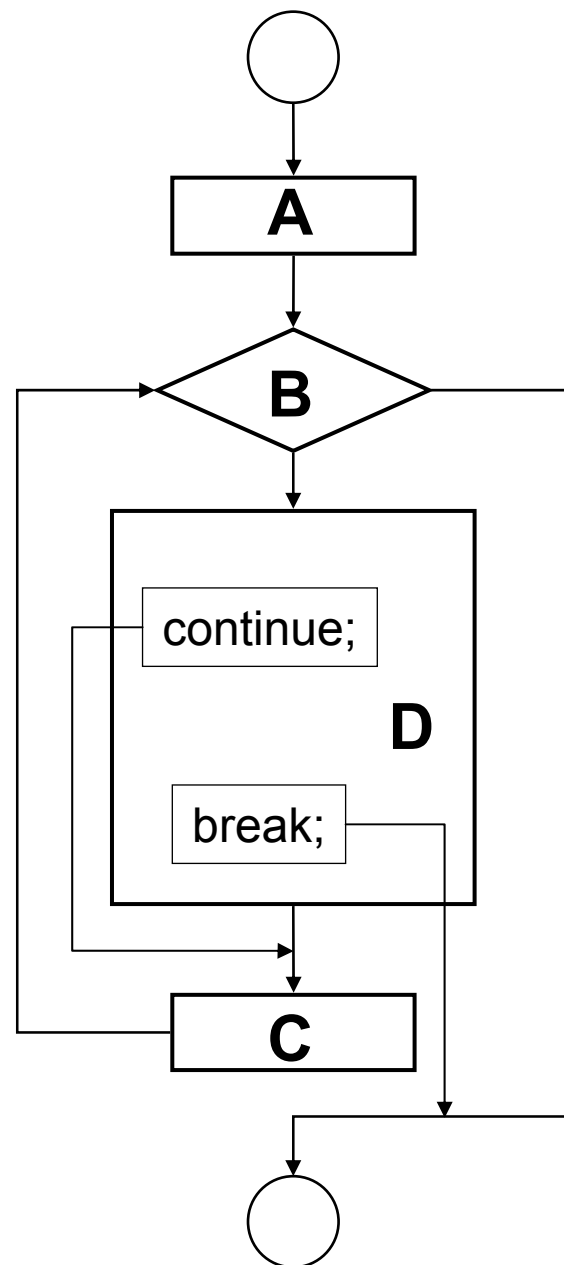
```
#include <stdio.h>
int main()
{
    char c;
    while ((c = getchar()) != EOF)
    {
        if(c >= '0' && c <= '9')
            continue;
        putchar(c);
    }
    return 0;
}
```

```
u65tf43d9i 765z
utfdi z
```

```
#include <stdio.h>
int main()
{
    int i;
    for(i=0; i<10; i++)
    {
        if(i<5)
            continue;
        printf("%d\n", i);
    }
    return 0;
}
```

```
5
6
7
8
9
```

```
• • •  
for(A ; B ; C)  
{  
    • • •  
    break;  
    • • •  
    continue;  
    • • •  
}  
• • •
```

**D**



# Избор од повеќе можности

## `switch - case`

```
switch(izraz)  
{  
    case konstanta1: blok_naredbi1;  
    case konstanta2: blok_naredbi2; break;  
    . . .  
    case konstantan: blok_naredbin;  
    default:        naredbi;  
}
```

*izraz* мора да резултира во `int` или `char` тип.

- Не смее да има два или повеќе case изрази со иста вредност. Програмата продолжува со наредбите зад case наредбата со вредноста на пресметаниот израз од switch наредбата. Со извршувањето на следните наредби се продолжува сè додека не се најде на наредбата break или до крајот на switch-case блокот.
- Доколку нема case наредба со соодветна вредност се извршуваат наредбите од default блокот. Ако не е наведен default блок на наредби, не се случува ништо - се продолжува со наредбите зад switch-case блокот.

```
#include <stdio.h>
int main()
{
    char c;
    c = getchar();
    switch (c) {
        case '1': printf("eden");      break;
        case '2': printf("dva");      break;
        case '3': printf("tri");      break;
        case '4': printf("cetiri");   break;
        case '5': printf("pet");      break;
        case '6': printf("sest");     break;
        case '7': printf("sedum");    break;
        case '8': printf("osum");     break;
        case '9': printf("devet");    break;
        case '0': printf("nula");     break;
        default: printf("ne e cifra");
    }
    return(0);
}
```

Што ќе отпечати следнава програма?

```
#include<stdio.h>
int main()
{   int j = 0;
    while ( j < 6 )
    {
        switch ( j )
        {
            case 0: j++;
            case 1: j++;      break;
            case 2:
            case 3: j += 2;   break;
            default: j = j - 1;
        }
        printf("Vrednosta na j e %d\n", j++);
    }
    return(0);
}
```

```
Vrednosta na j e 2
Vrednosta na j e 5
```

# Наредба **goto**

Основниот облик на оваа наредба е следниот

```
goto (ime_na_oznaka) ;
```

со тоа што некаде во програмата го имаме името на  
ознаката во облик

```
ime_na_oznaka :
```

Контролата на програмата по извршувањето на оваа  
наредба се пренесува на наредбата со ознаката.

Да се состави програма што ќе ги отпечати сите прости броеви помали од даден број.

```
#include<math.h>
#include<stdio.h>
int main()
{
    int i,j,k,n,x = 0;
    printf("Vnesi broj ");
    scanf("%d",&n);
    printf("Prosti broevi pomali od %d se\n",n);
    for(i = 1; i < n; i+=2)
    {
        k = 1;
        for(j = 2; j <= sqrt(i); j++)
            if(i%j == 0)
            {
                k = 0; break;
            }
        if(k)
        {
            printf("%d ",i);    x++;
        }
    }
    printf("\n Vkupno %d prosti broevi",x);
    return(0);
}
```

Програма која пресметува вредност на едноставен аритметички израз  
(без приоритети, само цели броеви):

```
#include <stdio.h>
int main()
{
    char operator = '+';
    int broj, resenie = 0;
    do
    {
        scanf("%d",&broj);
        switch(operator)
        {
            case '+': resenie += broj; break;
            case '-': resenie -= broj; break;
            case '*': resenie *= broj; break;
            case '/': resenie /= broj; break;
        }
    }
    while((operator = getchar()) != '=');
    printf("Resenieto e %d",resenie);
    return 0;
}
```

$15+2*2-4/3+1=$   
Resenieto e 11