

Контрола на текот на програмата (while, do...while, switch)

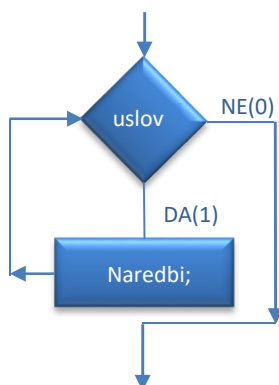
1. Структури за повторување, циклуси (while и do...while)

While структурата за повторување е една од трите структури за повторување во C и воедно наједноставна. Овозможува повторување на една или повеќе наредби.

Слично како и кај if структурата, така и во while структурата се задава услов (**задолжително во мали загради ()**) што се евалуира, односно проверува. Наредбата (наредбите) од циклусот се извршуваат се додека условот од while структурата е точен (враќа ненулта вредност). Повторувањето на наредбите од циклусот завршува кога условот ќе врати неточна вредност (нула) и програмата продолжува со извршување на следната наредба од телото на програмата (што не е дел од while структурата). Доколку треба да се изврши само една наредба, таа едноставно се пишува по while делот, доколку, пак, треба да се извршат повеќе наредби (блок наредби) по while делот, тие се ставаат во блок ограничен со големи загради { }.

while (услов)

тело (наредба/наредби за повторување);



```
int j = 5;

while (j > 0)
    printf("j = %d\n", j--);
```

while (j > 0)	j = 5
{	j = 4
printf("j = %d\n", j);	j = 3
j--;	j = 2
}	j = 1

ВАЖНО!

Доколку по грешка се напише знакот ; по while делот со условот, наредбата/наредбите од while структурата стануваат дел **"не прави ништо"**. Бидејќи условот останува непроменет програмата влегува во бесконечна јамка (**loop**). Во ваков случај програмата нема да врати резултат, бидејќи е зафатена со **"неправењето ништо"**, поради што следните наредби по while структурата никогаш не се извршуваат.

Пример: int j = 5;

```
while (j>0); ←
    printf("j = %d\n", j--);
```

ЕКРАН

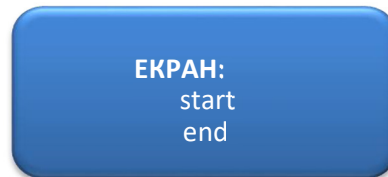
ВАЖНО!

Многу важно е да се запомни дека сите С услови се while услови, односно **“се додека е исполнет условот се извршуваат наредбите”**. Наредбите од циклусот while се извршуваат се додека е точен условот (вредност различна од 0).

Пример:

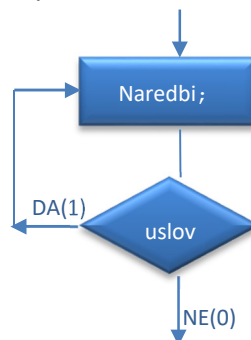
Во следниот пример, програмерот најверојатно имал намера наредбата од while структурата да се извршува се додека j не стане еднакво на 0. Сепак ова не е начинот да се постави условот, точен услов би бил се додека j е различно од 0 ($j \neq 0$).

```
int j = 5;
printf("start\n");
while (j == 0)
    printf("j = %d\n", j--);
printf("end\n");
```

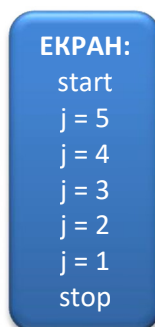


do. . . while структурата во С е “превртена” верзија на структурата за повторување while. Ако во while циклусот по условот следува телото со наредби што се повторуваат, кај do. . . while циклусот прво се пишува телото со наредби за повторување, па следува условот. Од ова произлегува дека наредбите од циклусот ќе бидат извршени сигурно еднаш, пред да се провери условот. Ако условот е неточен, наредбите од циклусот do. . . while нема да бидат повторно извршени.

do
тело
(наредба/наредби
за повторување);
while (услов);



```
int j = 5; printf("start\n");
do
    printf("j = %d\n", j--);
while (j > 0);
printf("stop\n");
```



```
int j = -10;
printf("start\n");
do {
    printf("j = %d\n", j);
    j--;
} while (j > 0);
printf("stop\n");
```



ВАЖНО!

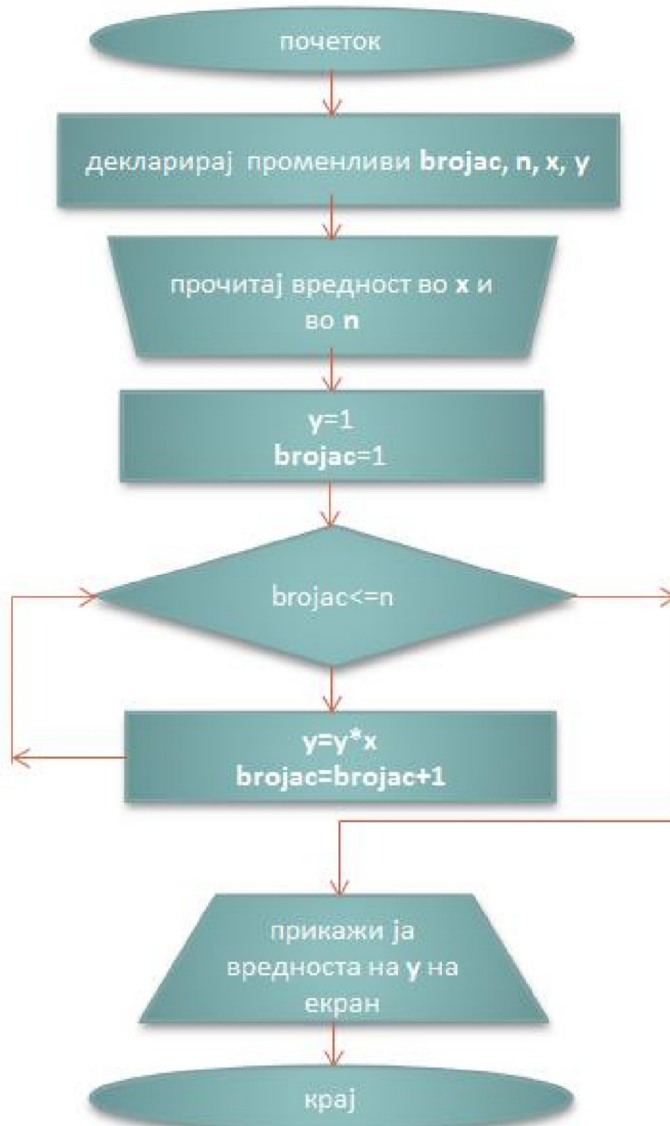
Да не се погреша и наместо == да се напише =.

Задачи:

1. Да се напише програма за пресметување на $y = x^n$ за даден природен број n и реален број x .

```
#include <stdio.h> int main ()
{
    int brojac, n;
    float x, y;
    printf("vnesi ja osnovata: ");
    scanf("%f", &x);
    printf("vnesi go eksponentot: ");
    scanf("%d", &n);
    y = 1; brojac = 1;
    while (brojac <= n)
    {
        y *= x;
        brojac++;
    }
    printf("%f^%d = %f\n", x, n, y);

    return 0;
}
```

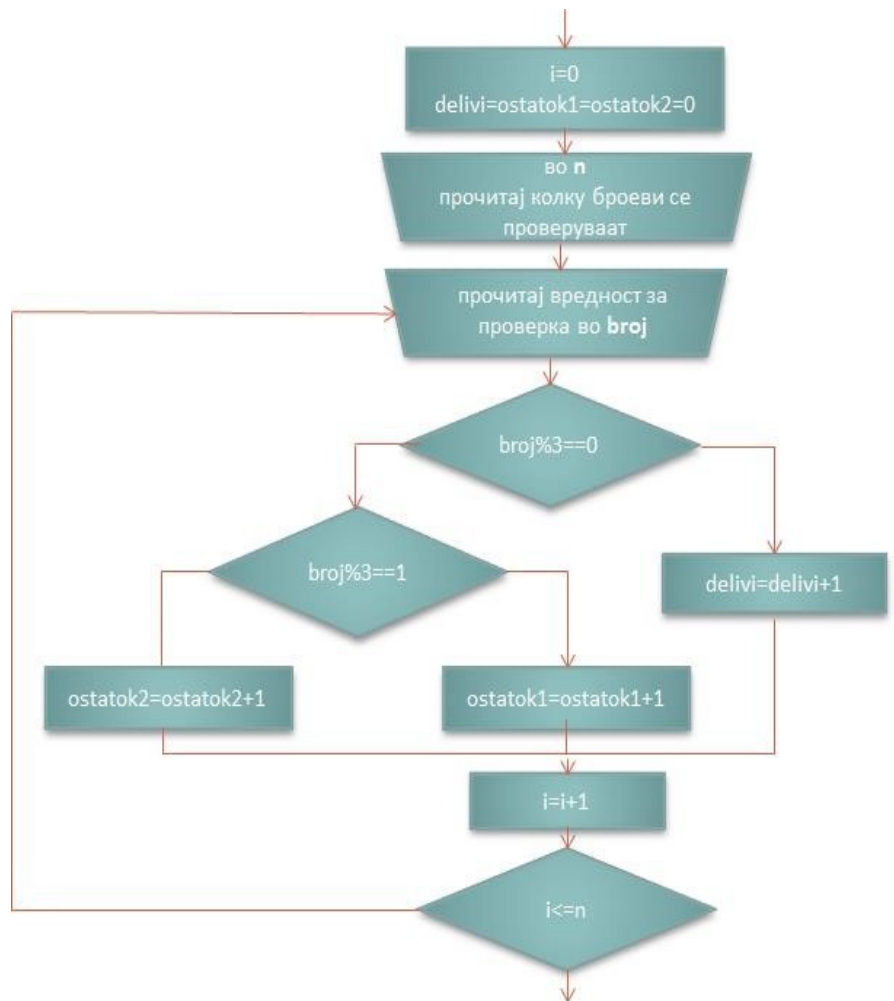


2. Да се напише програма која од n броеви (внесени од тастатура) ќе го определи бројот на броеви што се деливи со 3, при делењето со 3 имаат остаток 1, односно 2.

```
#include <stdio.h>
int main ()
{
    int n, i, broj, del, os1, os2; del = os1 = os2 = 0;
    printf("Kolku broevi treba da se proveruivat za delivost so 3?\n"); scanf("%d", &n);
    i = 1;
    do {
        printf("Vnesete broj za proverka: ");
        scanf("%d", &broj);
        if ( broj % 3 == 0) del++;
        else if ( broj % 3 == 1) os1++;
        else os2++;
        i++;
    }
    while (i <= n);

    printf("%d broj(a) se delivi so 3.\n", del);
    printf("%d broj(a) imaat ostatok 1, pri delenje so 3.\n", os1);
    printf("%d broj(a) imaat ostatok 2, pri delenje so 3.\n", os2);

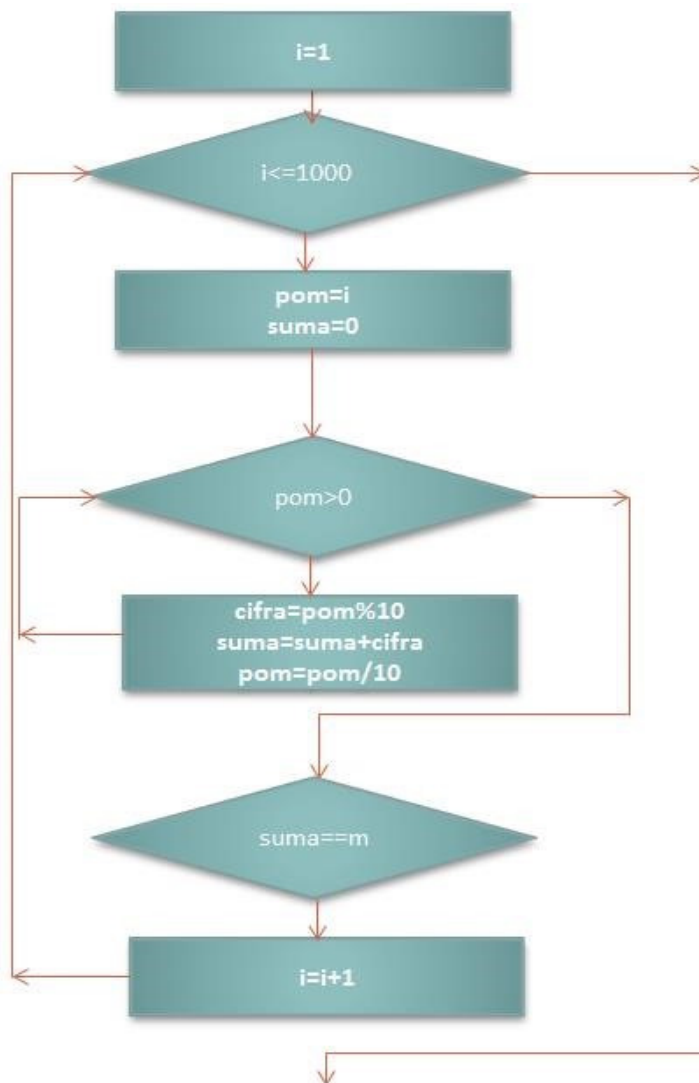
    return 0;
}
```



3. Да се напише програма која за даден природен број m ќе ги испише сите броеви помеѓу 1 и 1000 за кои сумата на цифрите изнесува m .

```
#include <stdio.h>
int main ()
{
    int m, i, j, pom, suma, cifra;
    printf("Vnesete go brojot m za sporedba: ");
    scanf("%d", &m);
    if (m > 27) printf("Ne postoi broj od 1 do 1000 so
    suma %d.\n", m);
    else {
        printf("Broevi cij zbir na cifri e %d se: \n", m);
        i = 1;
        while (i<=1000) {
            pom = i; suma = 0;
            while (pom > 0) {

                cifra = pom % 10;
                suma += cifra;
                pom /= 10;
            }//while
            if (suma == m) printf("%d\t", i);
            i++;
        } while
    } //else
    return 0;
}
```



ВАЖНО!

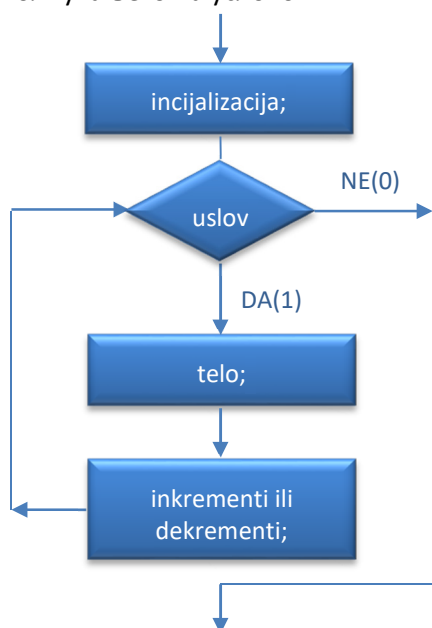
Да се нагласи зошто е потребно користењето на променливата **pom**.

2. Структура за повторување (for)

Од аспект на синтаксата **for** циклусот за повторување е најкомплициран. Сепак, сличен е на **while** циклусот, бидејќи и самиот содржи **while** тип на услов (се додека е исполнет, односно точен условот се извршуваат наредбите од телото).

for (иницијализација; услови; инкременти или декременти)
тело (наредба/наредби за повторување);

Во **for** делот содржани се почетните услови (иницијализација на променливи), услов/и за повторување, како и наредби за инкрементирање или декрементирање, од кои зависи исполнувањето на условот.



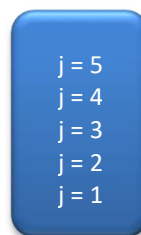
Наредбите од делот **иницијализација** се извршуваат точно еднаш, на почетокот - пред влезот во циклусот. Наредбите од делот **услови** се извршуваат пред почетокот на секој нов циклус и доколку резултираат со вредност која се интерпретира како логичка вистина се влегува во повторување на наредбите од циклусот, инаку се завршува повторувањето на циклусот и се продолжува со наредбите што следуваат по циклусот во програмата. Наредбите од делот **инкременти или декременти** се извршуваат на крајот на секој циклус (по извршувањето на сите наредби од **телото** на циклусот) по што се извршуваат наредбите од делот **услови** и доколку се задоволени, циклусот се повторува. Исто како и кај **if** и **while**, доколку треба да се изврши само една наредба, таа едноставно се пишува по **for** делот, доколку, пак, треба да се извршат повеќе наредби (блок наредби) по **for** делот, тие се ставаат во блок ограничен со големи загради **{ }**.

ВАЖНО!

Мора да се запомни дека знакот **;** мора да ги одделува трите дела на **for** наредбата. И условот во **for** наредбата мора да биде од типот **“се додека е исполнет условот”**.

```
int j;
```

```
for (j = 5; j > 0; j--)
    printf("j = %d\n", j--);
```



За разлика од програмскиот јазик PASCAL каде чекорот биде само 1, во C не постои ограничување за чекорот.

во for циклусот можеше да

Пример:

```
#include <stdio.h>
#include <math.h>
int main()
{
    double agol;
    for(agol = 0.0; agol < 3.14159; agol += 0.2)
        printf("sinus od %.1lf e %.2lf\n", agol, sin(agol));
    return 0;
}
```

Во овој случај променливата agol се зголемува за 0.2 при секое извршување на наредбата for, се додека е исполнет условот agol < 3.14159.

Во деловите за иницијализација и менување на вредности од for наредбата може да се наведат повеќе изрази по потреба одвоени со знакот запирка(.). Запирката гарантира секвенцијално извршување на наведените изрази.

Пример:

```
int i, j, k;
for (i = 0, j = 5, k = -1; i < 10; i++, j++, k--)
```

Ако нема потреба од наведување на изрази во овие два дела, тие можат да бидат изоставени, но **НЕ СМЕАТ** да се забораваат знаците ;.

```
for(; i < 10; i++, j++, k--)
```

наместо *for(; i < 10;)* подобро да се користи *while(i < 10) for(;;)* се креира бесконечна јамка и се чита “засекогаш”(for ever)

пример:

```
#define EVER (;;)
for EVER
{
    ...
}
```

3. Наредби **break** и **continue**

Од претходните примери за наредбите **while** и **for** се гледа можноста за креирање на бесконечен циклус. Знаејќи го ефектот од користењето на бесконечниот циклус произлегува дека треба да се одбегнува по секоја цена. Сепак, во C може да се користат бесконечни циклуси така што ќе се излегува, односно скокнува надвор од нив. Користењето на клучниот збор **break** овозможува излегување од било кој циклус, независно од условот и продолжување со првата следна наредба по циклусот.

Пример:

```
for(;;) { printf("vnesi int vrednost:");
    if(scanf("%d", &j) == 1) break;
    while((c = getchar()) != '\n')
        ;
}
```



ЕКРАН:

```
vnesi int vrednost: an int
vnesi int vrednost: no
vnesi int vrednost: 16
j = 16
```

Во овој случај, ако `scanf` врати 1 (односно се вчита преку тастатура цел број), се излегува од бесконечната јамка.

И додека **break** предизвикува моментално излегување од циклусот, клучниот збор **continue** предизвикува извршување на следната наредба од циклусот. Во случај на **while** и **do...while** циклусите, **continue** предизвикува директно реевалуирање, односно извршување на условот. Кај **for** циклусите пак, предизвикува извршување на делот за менување на вредностите на променливите од **for** наредбата, па реевалуирање на условот.

Пример:

```
for(j = 1; j <= 10; j++) {
    if(j % 3 == 0) continue;
    printf("j = %d\n", j);
}
```



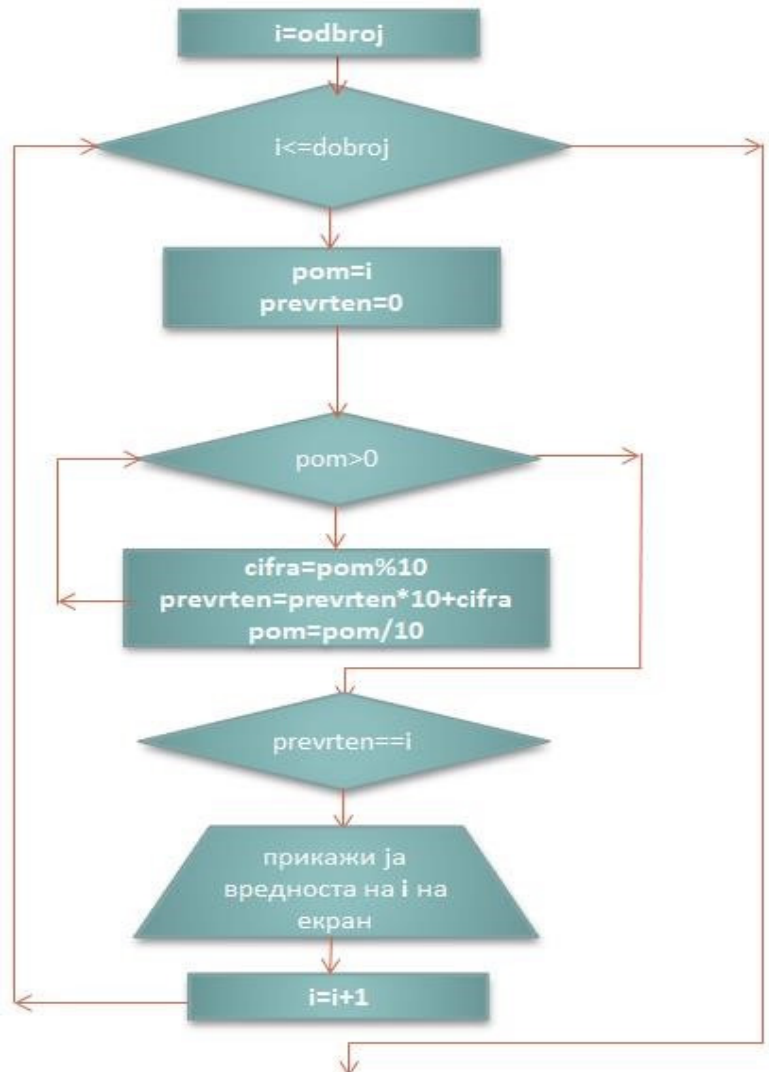
ЕКРАН:

```
j = 1
j = 2
j = 4
j = 5
j = 7
j = 8
j = 10
```


Задачи:

1. Да се напише програма која ќе ги испечати сите броеви од зададен опсег кои исто се читаат и од лево на десно и од десно на лево.

```
#include <stdio.h>
int main ()
{
    int i, odb, dob, pom, prev, cifra;
    printf("Vnesete vrednost za opsegot.\n");
    printf("Od koj broj?\n");
    scanf("%d", &odb);
    printf("Do koj broj?\n");
    scanf("%d", &dob);
    for (i = odb; i <= dob; i++) {
        pom = i;
        prev = 0;
        while (pom > 0) {
            cifra = pom % 10;
            prev = prev * 10 + cifra;
            pom /= 10;
        } //while
        if (prev == i) printf("%d\t", i);
    }
    return 0;
}
```



2. Да се напише програма која ќе ги испечати сите броеви помали од број N составени само од парни цифри и ќе врати колку такви броја има.

```
#include <stdio.h>

int main ()
{
    int i, n, pom, prov, cifra, brojac;
    printf("Do koj broj treba da se proveruva?\n");
    scanf("%d", &n); brojac = 0;
    for (i = 1; i <= n; i++) { pom=i;
        while ((pom > 0) ) {
            cifra = pom % 10;
            if (cifra % 2) break;
            pom /= 10;
        }//while
        if (pom==0) {
            printf("%d\t", i);
            brojac++;
        }//if
    }//for
    printf("\nIma vkupno %d broevi so samo parni cifri\n", brojac);
    return 0;
}
```

3. Да се напише програма која пресметува просек на еден студент, како и број на испити на кои студентот паднал. Бројот на испити што ги полагал студентот не е познат. Програмата завршува кога ќе се внесе вредност различна од број.

```
#include <stdio.h>

int main ()
{
    int i, ocena, suma, brocen, brpadnal;
    i = 1;
    suma = brocen = brpadnal = 0;
    printf("Vnesete bilo koja bukva za kraj!!!\n");
    for (;i;) {
        //citanje na ocena
        do {
            printf("Vnesete ocena :");
            i = scanf("%d", &ocena);
        } while ((ocena < 5 || ocena > 10) && i);
    }
```

```
if(!i) break; //se prochital znak razlichen od cifra
    //presmetuvanje za validna ocena pomegu 5 i 10
    if (ocena >= 6 && ocena <= 10 ) {
        suma += ocena;
        brocenii++;
    }
    else if (ocena == 5) brpadnal++;
} //for
//pecatenje na dobienite rezultati
if (brocenii)
    printf("Prosekot na studentot e %.2f.\n", (float)suma/brocenii);
else
    printf("Studentot nema polozeno ispit.\n");
if (brpadnal)
    printf("Studentot padnal na %d ispit(i).\n", brpadnal);
else
    printf("Studentot nema padnato na ispit.\n");

return 0;
}
```

4. Да се напише програма која за дадено n го пресметува збирот на првите n членови во редот: $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \dots$

```
int main() {
    int n,i;
    float zbir=0;
    printf("Vnesete cel broj");
    scanf("%d", &n);
    for(i=1;i<=n;i++)
        zbir+=(1.0/i);
    printf("Vrednosta na izrazot e %f",zbir);
    return 0;
}
```

Контрола на текот на програмата (switch)

1. Switch структура

Структурата switch има слична употреба како низа на вгнездени if/else наредби. Општиот облик на структурата е следниот:

```
switch ( izraz ) { case
konstanta1 :
    naredbi1;
    ....
    break ;
case konstanta2 :
    naredbi2;
    ....
    break ; default:
    naredbid;
    .... break ;
case konstantan :
    naredbin;    ..
    .. break ;
}
```

Со оваа структура, најпрво, се разрешува изразот напишан во заградите веднаш по зборчето **switch**, и потоа се прескокнува кон соодветната **case** константа (лабела) чија вредност е еднаква на вредноста на изразот. Не е дозволено да постојат дупли **case** константи. Вредноста на изразот *izraz* мора да биде цел број (int), знак (char) или енумерички тип. Константите (лабелите) во **case**, може да бидат наредени по кој било редослед, но **мора** да бидат константи. Доколку во **case** константите не постои вредност еднаква со вредноста на изразот, се извршуваат наредбите по лабелата **default**. Во случај да не постои **default** лабела, тогаш **switch** наредбата не извршува **ништо**. **Default** лабелата може да се наоѓа каде било во листата на лабели од **switch** наредбата, односно не мора да биде на крајот, но почитливо е ако се напише на крај.

Иако **default** не мора да се напише, понекогаш е zgodno во секоја switch структура да постои и оваа лабела, дури и доколку наредбите во неа не прават ништо.

```
default:
/* Ne pravi nishto */ break;
```

Доколку по наредбите што одговараат на соодветната **case** лабела не се напише зборчето **break**, тогаш С продолжува со извршување на наредбите од следната **case** лабела, иако тоа не би било логично. **Break** овозможува **завршување** на наредбата **switch** и продолжување со следната наредба од програмата по неа. Затоа, **break** не смее да се заборави, освен доколку програмерот, има таква намера.

```
control = 0;
/* losh primer na programerska praksa */
switch (control) { case 0:
printf("Reset\n"); case 1:
    printf("Initializing\n");
    break; case 2:
printf("Working\n");
}
```

Доколку, на овој начин се напишат наредбите во switch структурата, тогаш ако вредноста на control == 0, програмата ќе испечати

Reset

Initializing

затоа што case 0 не завршува со break. По печатењето на Reset, програмата **продолжува** со извршување на наредбите од следната лабела (case 1) и печати Initializing.

Задачи:

1. Да се напише програма што ќе работи како едноставен калкулатор: од тастатура ќе чита оператор и два броја и ќе ја изврши наведената операција. Задачата да се реши најпрво со if, а потоа со switch структура.

СО IF:

```
#include <stdio.h>
int main()
{
    char operator;
    float br1, br2, result=0;
    printf("Vnesete operator ('+', '-', '*', '/):");
    scanf("%c", &operator);
    printf("Vnesete dva broja:"); scanf("%f %f", &br1, &br2);

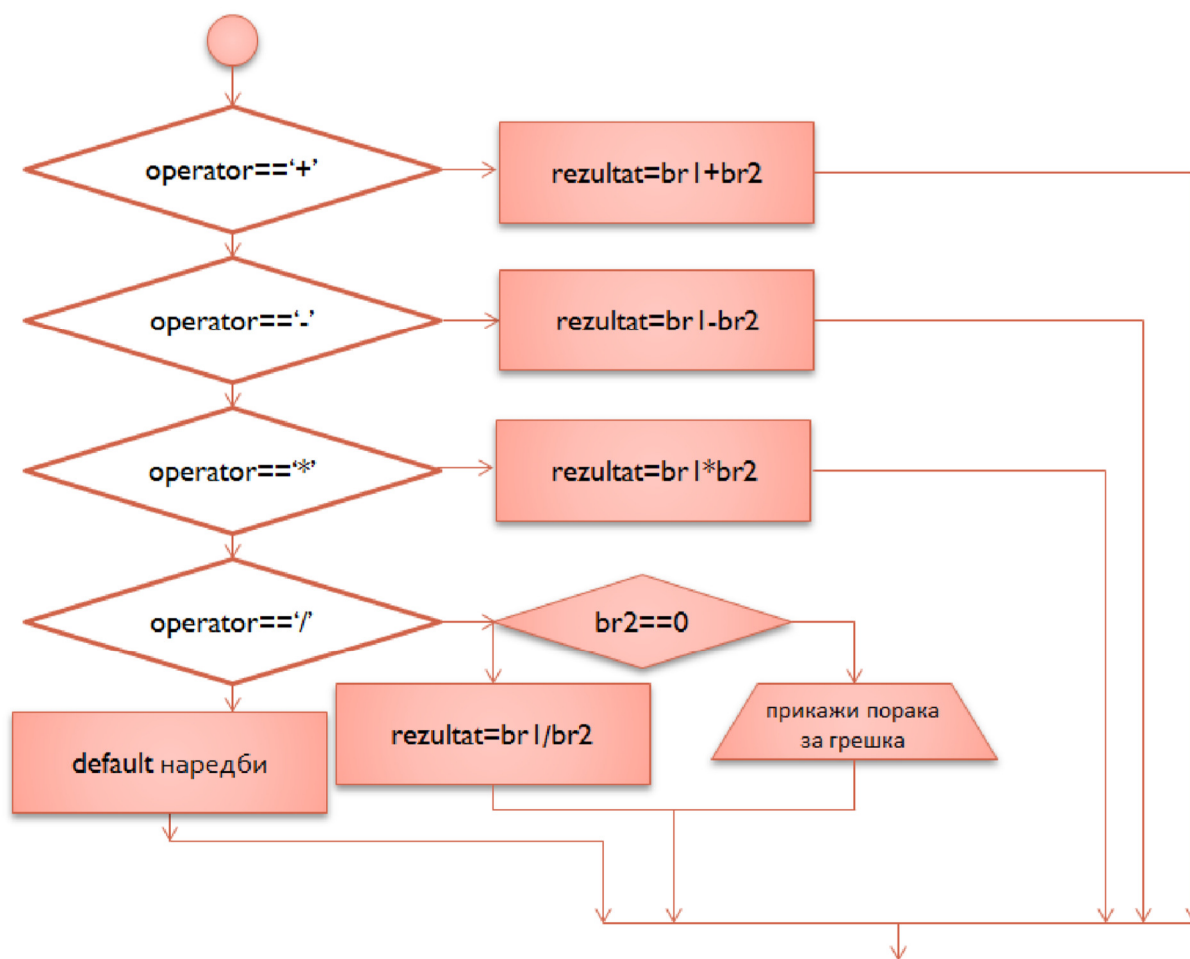
    if (operator == '+') {
        result = br1 + br2;
        printf("Rezultatot od operacijata: %.2f %c %.2f = %f", br1, operator, br2, result);
    }
    else if (operator == '-') {
        result = br1 - br2;
        printf("Rezultatot od operacijata: %.2f %c %.2f = %f", br1, operator, br2, result);
    }
    else if (operator == '*') {
        result = br1 * br2;
        printf("Rezultatot od operacijata: %.2f %c %.2f = %f", br1, operator, br2, result);
    }
    else if (operator == '/') { if (br2 == 0) {
        printf("Greshka: Delenje so 0\n");
    }
    }
```

```

        printf(" operacijata ke se ignorira\n");
    }
    else {
        rezult = br1 / br2;
        printf("Rezultatot od operacijata: %.2f %c %.2f = %f", br1, operator, br2, rezult);
    }
}
else printf("Nepoznat operator %c\n", operator);
return 0;
}

```

CO SWITCH:



```

#include <stdio.h>
int main()
{
    char operator;
    float br1, br2, rezult=0;
    printf("Vnesete operator ('+', '-', '*', '/'):");
    scanf("%c", &operator);
    printf("Vnesete dva broja:");

```

```

scanf("%f %f", &br1, &br2);

switch (operator) {
case '+': rezult = br1 + br2;
        printf("Rezultatot od operacijata: %.2f %c %.2f = %f", br1, operator, br2, rezult);
        break;
case '-': rezult = br1 - br2;
        printf("Rezultatot od operacijata: %.2f %c %.2f = %f", br1, operator, br2, rezult);
        break;
case '*': rezult = br1 * br2;
        printf("Rezultatot od operacijata: %.2f %c %.2f = %f", br1, operator, br2, rezult);
        break;
case '/':
        if (br2 == 0) {
            printf("Greshka: Delenje so 0\n");
            printf(" operacijata ke se ignorira\n");
        }
        else {
            rezult = br1 / br2;
            printf("Rezultatot od operacijata: %.2f %c %.2f = %f", br1, operator, br2, rezult);
        }
        break;
default: printf("Nepoznat operator %c\n", operator);
        break;
}
return 0;
}

```

2. Да се напише програма што ќе овозможи претворање на броевите во зборови на следниот начин: За 89 како излез ќе се добие “osum devet”. Дозволено е да се внесат **само двоцифрени** броеви.

```

#include <stdio.h>
int main()
{
    int broj,mala,golema;
    printf("Vnesete dvocifren broj:");
    scanf("%d", &broj);
    mala = broj % 10;
    golema = broj/10;

    switch (golema) {
    case 0: printf("nula ");
            break;
    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;
default: break;
}
switch (mala) {
case 0: printf("nula\n");
        break;
case 1: printf("eden\n");
        break;
case 2: printf("dva\n");
        break;
case 3: printf("tri\n");
        break;
case 4: printf("cetiri\n");
        break;
case 5: printf("pet\n");
        break;
case 6: printf("sest\n");
        break;
case 7: printf("sedum\n");
        break;
case 8: printf("osum\n");
        break;
case 9: printf("devet\n");
        break;
default: break;
}
return 0;
}
```


3. Доколку е познато дека бројот 85 се изговара како “osumdeset i pet”, да се промени претходната програма така што ќе може за броевите **од 0 до 100**, да печати текст, онака како што се изговараат броевите. На пример: за 13 ќе отпечати “trinaeset”, а за 100 ќе отпечати “sto”.

```
int main()
{
    int i;
    for (i=0;i<=100;i++)
    {
        gotovo=0; okruglo=0;
        if (i>9)
        {
            if (i==100) {printf("sto\n"); gotovo=1;}
            else
            {
                if ((i/10)==1) /* znaci vo intervalot [10-19] */
                {
                    gotovo=1;
                    switch (i)
                    {
                        case 10: printf("deset\n"); break;
                        case 11: printf("edinaeset\n"); break;
                        case 12: printf("dvanaeset\n");
                                break;
                        case 13: printf("trinaeset\n"); break;
                        case 14: printf("cetirinaeset\n");
                                break;
                        case 15: printf("petnaeset\n"); break;
                        case 16: printf("sesnaeset\n");
                                break;
                        case 17: printf("sedumnaeset\n");
                                break;
                        case 18: printf("osumnaeset\n");
                                break;
                        case 19: printf("devetnaeset\n");
                                break;
                    } //switch
                } //if(i/10)
            } /* znaci vo intervalot [20-99] */
            {
                switch (i/10)
                {
                    case 2: printf("dvaeset");
                            break;
                    case 3: printf("trieset");
                            break;
                    case 4: printf("chetirieset");
```

```

        break;
    case 5: printf("pedeset");
        break;
    case 6: printf("seeset");
        break;
    case 7: printf("sedumdeset");
        break;
    case 8: printf("osumdeset");
        break;
    case 9: printf("devedeset");
        break;
    } /* switch */
    if (i%10==0) { okruglo=1; printf("\n"); }
    } /* kraj na else za if (i/10==1) */
    } /* kraj na else za if (i==100) */
} /* kraj na if (i>9) */
if (!gotovo)
{ if ((i>20) && (!okruglo)) printf(" i ");
  switch (i%10)
  {
    case 0: if (!okruglo) printf("nula \n");
            break;
    case 1: printf("eden\n");
            break;
    case 2: printf("dva\n");
            break;
    case 3: printf("tri\n");
            break;
    case 4: printf("cetiri\n");
            break;
    case 5: printf("pet\n");
            break;
    case 6: printf("sest\n");
            break;
    case 7: printf("sedum\n");
            break;
    case 8: printf("osum\n");
            break;
    case 9: printf("devet\n");
            break;
  } /* switch */
} /* kraj na if (!gotovo) */
} /* kraj na for */
} /* kraj na main */

```