Универзитет “Св. Кирил и Методиј” во Скопје
Факултет за електротехника и информациски технологии

ПРОГРАМИРАЊЕ И АЛГОРИТМИ

КОНТРОЛА НА ТЕК

- Програмски јазик C –

Програмирање и алгоритми

КОНТРОЛНИ СТРУКТУРИ

КОНТРОЛНИ СТРУКТУРИ

- Секвенцијално извршување
  - Наредби се извршуваат една по друга по редослед на пишување
- Предавање на контролата
  - Кога следната наредба што се извршува не е онаа што е следно напишана
- Теорема за правилно програмирање (Bohm и Jacopini)
  - Сите програми може да се напишат со 3 контролни структури
    - **Секвенцијална структура:** Вградена во C. Програмите по дефиниција се извршуваат секвенцијално (наредба по наредба)
    - **Структури за избор:** C има три типа: `if`, `if/else` и `switch`.
    - **Структури за повторување:** C има три типа: `while`, `do/while` и `for`

ПРОБЛЕМ СО ПРИМЕНА НА СТРУКТУРА ЗА ИЗБОР

Проблем: Да се одреди дали еден студент го положил курсот Програмирање и алгоритми. Студентот **положил** **доколку** **средната вредност на поените** од **2 парцијални испити** што се **внесуваат од тастатура** е **поголема од 50 поени**.

Нов тип на проблем. Треба да се донесе одлука.

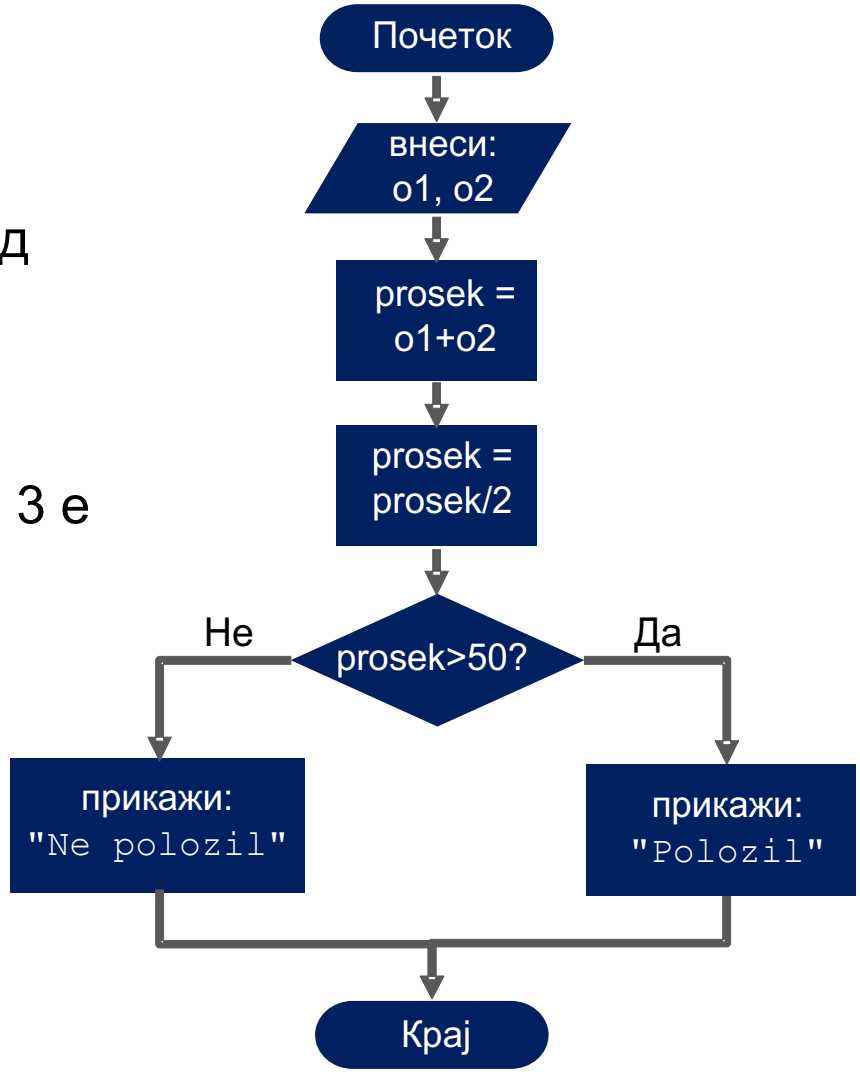
Чекори при решавањето:

1. Внеси ги двете парцијални оцени од тастатура
2. Пресметај ја нивната сума
3. Сумата подели ја со 2
4. Доколку средната вредност (резултатот од чекор 3) е поголема од 50, на екранот прикажи „Polozhil“, инаку на екранот прикажи „Ne polozhil“

АЛГОРИТАМ

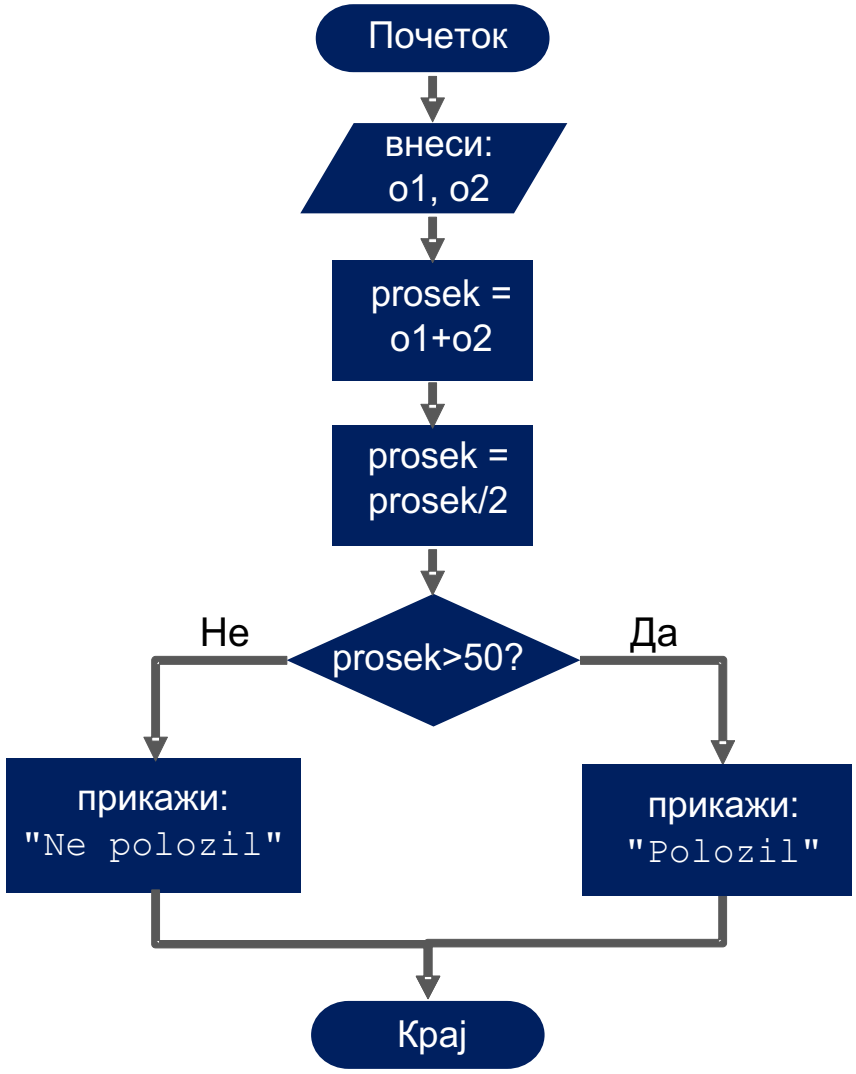
Постапка изразена со чекори:

1. Внеси ги двете парцијални оцени од тастатура
2. Пресметај ја нивната сума
3. Сумата подели ја со 2
4. Провери **дали** резултатот од чекор 3 е поголем од 50
   a) Ако одговорот е ДА на екранот прикажи “Polozil".
      Оди на чекор 5.
   b) Ако одговорот е **НЕ** на екранот прикажи “Ne polozil".
      Оди на чекор 5.
5. Крај



ПРОГРАМА

```
#include <stdio.h>
int main(){
    int o1, o2, prosek;
    scanf("%d%d",&o1,&o2);
    prosek = o1+o2;
    prosek = prosek/2;
    if(prosek>50)
        printf("Polozil\n");
    else
        printf("Ne polozil\n");
    return 0;
}
```



СТРУКТУРА ЗА ИЗБОР IF (ОД ЕДНА МОЖНОСТ)

- `if` структура за избор:
  - Да се изврши или не текот на активности опфатен во `if` структурата
- Синтакса на `if` структурата:

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

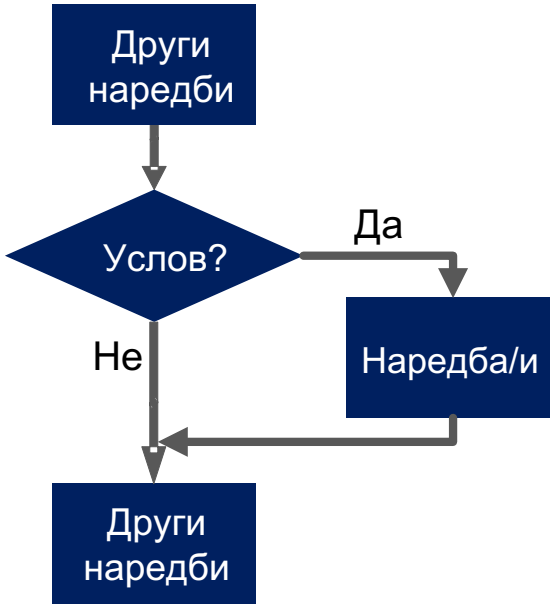
```
if (uslov){
  blok_naredbi_za_vistinit_uslov;
}
```

  - Ако условот е **точен** (`true`) се извршува наредбата во `if` структурата, а потоа се продолжува на следната наредба (по `if` структурата).
  - Ако условот е **неточен** (`false`) наредбата во `if` структурата се игнорира (прескокнува) и се извршува следната наредба (по `if` структурата).

```
if(prosek>50)
    printf("Polozil\n");
```

СТРУКТУРА ЗА ИЗБОР IF (ОД ЕДНА МОЖНОСТ)

- Симбол за одлука – треба да се направи одлука
  - содржи израз кој може да биде вистина (`true` **– да има ненулта вредност**) или невистина (`false` **– да има вредност 0**)
- Толкување
  - Провери го условот
  - Одреди ја патеката по која следно ќе се движи програмата



Резултат од Услов?:
Нула (zero) - `false`
Различно од нула (nonzero) – `true`

Пример:
`3 – 5` е `true`
`-1 + 1` е `false`
`0` е `false`
`100` е `true`

СТРУКТУРА ЗА ИЗБОР IF/ELSE (ОД ДВЕ МОЖНОСТИ)

- `if/else` структура за избор
  - Се избира еден од (два) можни тека на активности: едни активности ако Услов? е вистина (`true`), а други активности ако Услов? е невистина (`false`)
- Синтакса:

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

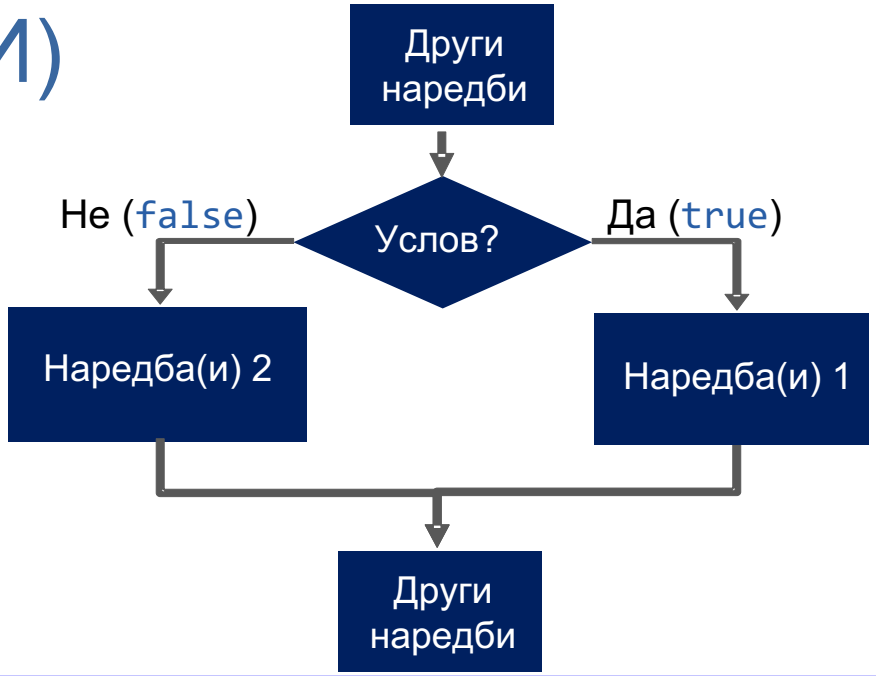
```
if (uslov){
  blok_naredbi_za_vistinit_uslov;
}
else {
  blok_naredbi_za_nevistinit_uslov;
}
```

- Пример:

```
if(prosek>50)
    printf("Polozil\n");
else
    printf("Ne polozil\n");
```

СТРУКТУРА ЗА ИЗБОР IF/ELSE (ОД ДВЕ МОЖНОСТИ)

- Најчеста грешка:
  - Кога има повеќе наредби да не се користат загради



Погрешно користење:

```
if(prosek>50)
    printf("Polozil\n");
else
    printf("Ne polozil\n");
    printf("Povtorno da polaga!");
```

Наредбата:

```
printf("Povtorno da polaga!");
```

Ќе се изврши независно од тоа дали `prosek>50`

Правилно користење:

```
if(prosek>50)
    printf("Polozil\n");
else {
    printf("Ne polozil\n");
    printf("Povtorno da polaga!");
}
```

ВГНЕЗДЕНИ IF/ELSE СТРУКТУРИ

- Проверка на повеќе услови со вгнездување `if/else` структури во `if/else` структури (длабоко вгнездување се избегнува)

```
if(prosek>90)
    printf("Ocenka 10\n");
else if(prosek>80)
        printf("Ocenka 9\n");
     else if(prosek>70)
             printf("Ocenka 8\n");
          else if(prosek>60)
                  printf("Ocenka 7\n");
               else if(prosek>50)
                       printf("Ocenka 6\n");
                    else
                       printf("Ocenka 5\n");
```

- Кога ќе се исполни условот, останатите наредби се прескокнуваат

ПРАКТИЧНИ СОВЕТИ ПРИ КОРИСТЕЊЕ НА IF/ELSE СТРУКТУРИ

- Секој `else` се однесува на првиот слободен `if` над него независно од позицијата на поставувањето на кодот
  - За поврзување на `else` со правилното `if` треба да се користат големи загради { }
- Пример:

Неправилно:

```
if(prosek>50)
    if(prosek<100)
       printf("Polozil\n");
else
    printf("Ne polozil\n");
```

Правилно:

```
if(prosek>50) {
    if(prosek<100)
       printf("Polozil\n");
}
else
    printf("Ne polozil\n");
```

- Често се пишува `if(izraz)` наместо `if(izraz != 0)` бидејќи со самото тоа што `izraz` е различно од нула тоа е `true`
- Слично се пишува `if(!izraz)` наместо `if(izraz == 0)`

ПРАКТИЧНИ СОВЕТИ ПРИ КОРИСТЕЊЕ НА IF/ELSE СТРУКТУРИ

- Под `if` или `else` доколку не се стават загради тогаш само една наредба со поврзува со нив – доколку станува збор за друга `if/else` структура, тогаш целата структура се смета за една наредба.
  - Пример:

```
if(sredstva>cena)
    printf("Kupi\n");
else
    if(imash_prijatel)
        printf("Pozajmi i kupi\n");
    else
        printf("Zaraboti i kupi\n");
```

- Во изразот во `if` наредбата може да се стават и посложени изрази, наредби за доделување, оператори и повикување на функции
  - Примери:

```
if(c=='a'||c=='e'||c=='i'||c=='o'||c=='u')
    printf("Promenlivata c e samoglaska\n");
```

```
if((c=getchar())!='\n')
```

ПРОБЛЕМ СО ПРИМЕНА НА СТРУКТУРА ЗА ПОВТОРУВАЊЕ

Проблем: Да се одреди аритметичка средина на броевите од 1 до 1000.
Решение: (1+2+3+4+5+...+1000)/1000

Нов тип на проблем. Треба да се повторува операцијата собирање одреден број пати (во случајот 1000).

Чекори при решавањето:

1. На почеток сумата постави ја на вредност 0.
2. Постави i=1
3. **Ако i≤1000** оди на чекор 4, инаку оди на чекор 7 (крај на повторување)
4. На сумата додај го i.
5. Зголеми го i за 1.
6. Оди на чекор 3.
7. Пресметаната сума подели ја со 1000.
8. Прикажи средна вредност
9. Крај

Програмирање и алгоритми

ЦИКЛУСИ

ЦИКЛУСИ

- Компјутерите се наменети за апликации во кои операциите се повторуваат многу пати.

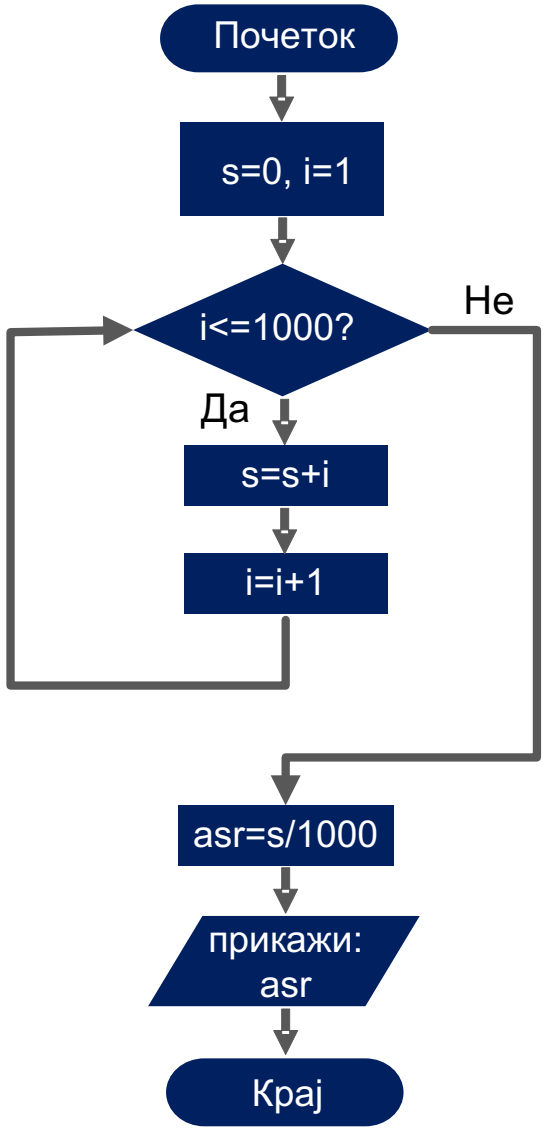
- Ако иста задача се повторува повторно и повторно може да се искористи јамка за поедноставување на големината и комплексноста на програмата

- Јамка/Циклус
  - Група инструкции кои компјутерот ги извршува повторно, се додека некој услов останува вистинит

АЛГОРИТАМ

Постапка изразена со чекори:

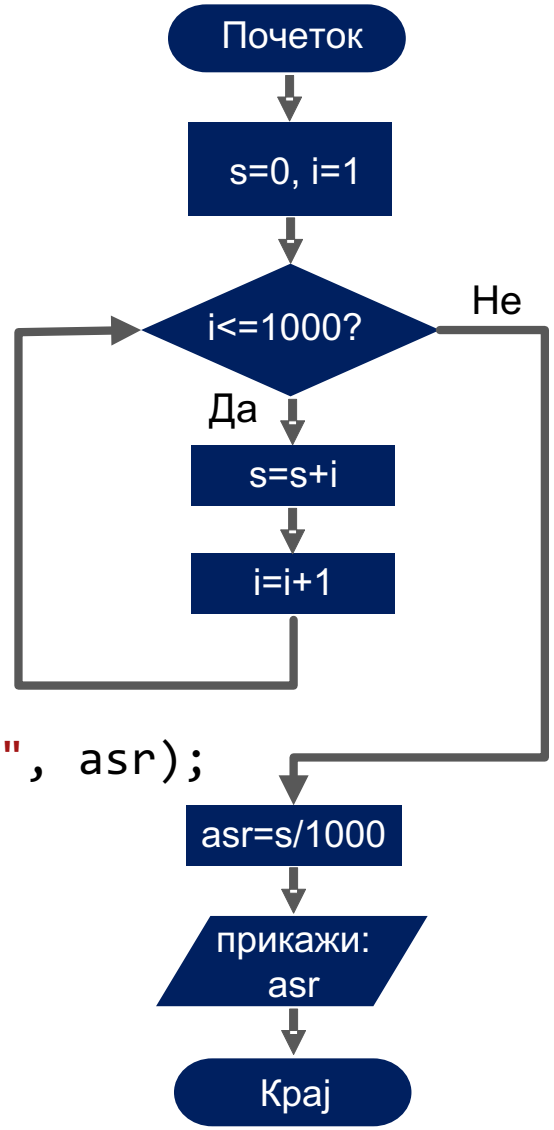
1. На почеток сумата постави ја на вредност 0.
2. Постави i=1
3. **Ако i≤1000** оди на чекор 4, инаку оди на чекор 7 (крај на повторување)
4. На сумата додај го i.
5. Зголеми го i за 1.
6. Оди на чекор 3.
7. Пресметаната сума подели ја со 1000.
8. Прикажи средна вредност
9. Крај



ПРОГРАМА

```
#include <stdio.h>
int main(){
    int i=1, suma = 0;
    float asr;
    while(i<=1000){
        suma = suma + i;
        i = i +1;
    }

    asr = (float)suma/1000;
    printf("Aritmetichnata sredina e %.2f\n", asr);
    return 0;
}
```



СТРУКТУРА ЗА ПОВТОРУВАЊЕ WHILE

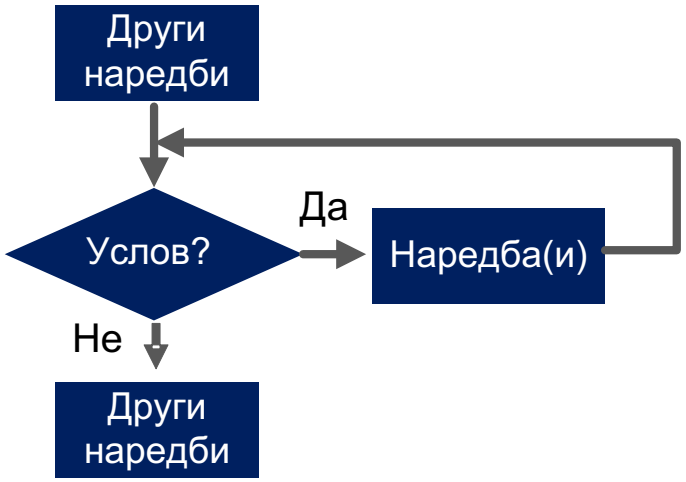
- Се дефинира акција што ќе се повторува се додека некој услов останува точен (`true`)
- Синтакса:

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

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

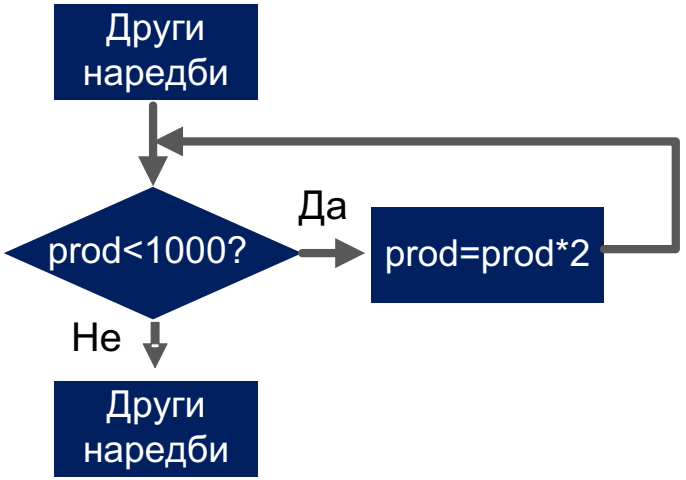
- `while` јамка се повторува се додека условот е `true`
- Кога условот ќе стане `false`, јамката (циклусот) завршува
- **Условот се испитува на почетокот (уште пред влезот во циклусот).** Наредбите од циклусот се повторуваат **ниту еднаш** (ако условот не е исполнет уште првиот пат) или **барем еднаш** (ако условот е исполнет првиот пат)

СТРУКТУРА ЗА ПОВТОРУВАЊЕ WHILE



Пример: Пресметај го најмалиот број кој е степен на 2 и е поголем од 1000.

```
#include <stdio.h>
int main(){
    int prod=2;
    while(prod<1000)
        prod = prod*2;
    printf("Brojot e %d\n", prod);
    return 0;
}
```



ПРИМЕР ЗА КОРИСТЕЊЕ НА WHILE ЦИКЛУС

Да се напише програма која ќе дозволи внесување на 5 цели броеви од тастатура и ќе ја прикаже нивната средна вредност.

```
#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 e %f\n", (float)suma/(n-1));
    return 0;
}
```

СТРУКТУРА ЗА ПОВТОРУВАЊЕ DO/WHILE

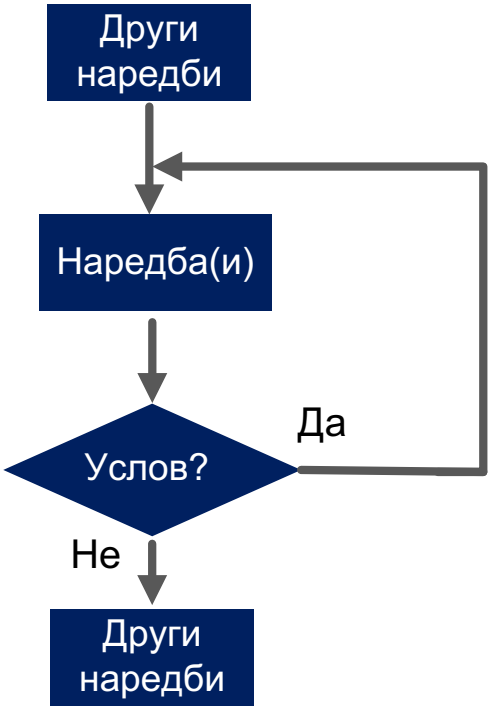
- Слична на `while` но услов за повторување се проверува **откако** ќе се изврши телото на јамката
- Синтакса:

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

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

- **Условот се испитува на крајот, поради што блокот на наредби се извршува најмалку еднаш!** Кај `while` услов за повторување се проверува пред да се изврши телото на јамката па за `while` циклусот може да се случи телото на јамката да не се изврши ниеднаш

СТРУКТУРА ЗА ПОВТОРУВАЊЕ DO/WHILE



Пример: Печати ги сите цели броеви од 1 до 10

```
#include <stdio.h>
int main(){
    int counter=1;
    do {
        printf("%d ",counter);
        counter++;
    } while(counter<=10);
    return 0;
}
```

КОНЦЕПТИ НА ПРОГРАМИРАЊЕ СО ЦИКЛУСИ

- Јамка контролирана со бројач
  - Конечно повторување – се знае колку пати ќе се изврши јамката
  - Контролна променлива за броење колку пати ќе се изврши јамката

- Јамка контролирана со „чувар“ (знаменце)
  - Бесконечно повторување
  - Се користи кога бројот на повторувања не е познат
  - **Вредноста на „чуварот“** ја содржи информацијата за тоа кога се случува моментот на **„крај на податоците“**

ЈАМКА КОНТРОЛИРАНА СО БРОЈАЧ

- За оваа јамка се потребни
  - **Име за контролната променлива** (бројач за јамка).
  - **Иницијална вредност** за контролната променлива.
  - **Услов** кој проверува дали се работи за **последната вредност** на контролната променлива (дали да продолжи јамката или не).
  - **Инкремент (или декремент)** за кој се менува контролната променлива секој пат кога ќе помине низ јамката.
- Јамката се повторува се додека **бројачот не стигне до дадена вредност**.
- **Конечно повторување**: бројот на повторувања е познат

ЈАМКА КОНТРОЛИРАНА СО БРОЈАЧ

Пример: Печати ги сите цели броеви од 1 до 10

```
#include <stdio.h>
int main(){
    int counter=1;       // initialization
    while(counter<=10){ //repetition condition
        printf("%d ",counter);
        ++counter;       //increment
    }
    return 0;
}
```

- `int counter=1;`
  - Дава име на контролната променлива `counter`,
  - Декларира дека е цел број (integer)
  - Резервира простор за него во меморијата
  - Го поставува на почетна вредност `1`

ЈАМКА КОНТРОЛИРАНА СО „ЧУВАР“

- Се користи вредност „чувар“ (sentinel)
  - Се вика и „Сигнална вредност“, „Dummy вредност“, „Знаменце вредност“
  - Означува **“крај на внес на податоци”**
  - Јамката завршува **кога ќе се внесе „чуварот“**
  - Вредноста на „чуварот“ се бира така што таа да не биде иста со регуларен влез (на пример -1)

Пример: Печати непознат број на букви од тастатура

Кој знак нема никогаш да се испечати?

```
#include <stdio.h>
int main(){
    char c;                              //deklaracija
    while ((c=getchar())!= 's'){         //uslov
       printf("Znakot e %c\n", c);
    }
    return 0;
}
```

ЈАМКА КОНТРОЛИРАНА СО „ЧУВАР“

Пример: Пресметај средна вредност на непознат број на цели броеви кои се внесуваат од тастатура, внесувањето да се прекинува кога ќе се внесе буква

```
#include <stdio.h>
int main() {
    int a;                                  //deklaracija
    int sum = 0, b, n = 0;
    while ((a=scanf("%d",&b))!= 0) {    //uslov
        sum = sum + b; n++;
    }
    printf("Srednata vrednost e %.2f\n", (float)sum/n);
    return 0;
}
```

- Функцијата `scanf("%d",&b)` враќа број на правилно прочитани променливи (ако внесеме буква `scanf("%d",&b)` функцијата ќе врати 0, а ако внесеме број функцијата ќе врати 1)

СТРУКТУРА ЗА ПОВТОРУВАЊЕ FOR

- Јамка со бројач (изброиво) – синтакса:

```
for(inicijalizacija ; uslovi ;  inkrementi/dekrementi)
        naredba;
```

```
for(inicijalizacija ; uslovi ;  inkrementi/dekrementi)
{
    blok_naredbi;
}
```

- Може да се запишат и како `while` јамка

```
inicijalizacija;
while(uslovi){
   naredbi;
   inkrementi/dekrementi;
}
```

СТРУКТУРА ЗА ПОВТОРУВАЊЕ FOR (II)

- **Делот за услови** содржи логички и релациски изрази (услови кои ќе го контролираат извршувањето на циклусот)
- Ако условот (во делот за услови) уште пред почеток на `for` јамката е `false`
  - Телото на `for` структурата воопшто не се извршува
  - Веднаш се извршува наредбата напишана после `for` структурата
- **Сè додека условот е исполнет** се повторуваат наредбите од циклусот и се инкрементираат/декрементираат бројачите

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

- Во **третиот дел** се инкрементираат/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)
```

СТРУКТУРА ЗА ПОВТОРУВАЊЕ FOR (III)

- Сите делови од `for` структурата

  `(inicijalizacija, uslovi i inkrementi/dekrementi)`
  може да содржат аритметички изрази

  - Пример: ако `x = 2 и y = 10`

  ```
  for ( j = x; j <= 4 * x * y; j += y / x )
  ```

  е еквивалентно со

  ```
  for ( j = 2; j <= 80; j += 5 )
  ```

- "Инкрементот" може да биде негативен (декремент)
- Во секој од овие делови може да се стават **повеќе** на број **произволни изрази** (кои може да содржат произволни наредби) притоа изразите се **раздвоени со запирка** и се извршуваат од лево кон десно (ова е важно кај **условот** кој **ја добива вредноста на најдесниот израз**)

ИЗВРШУВАЊЕ НА FOR НАРЕДБА

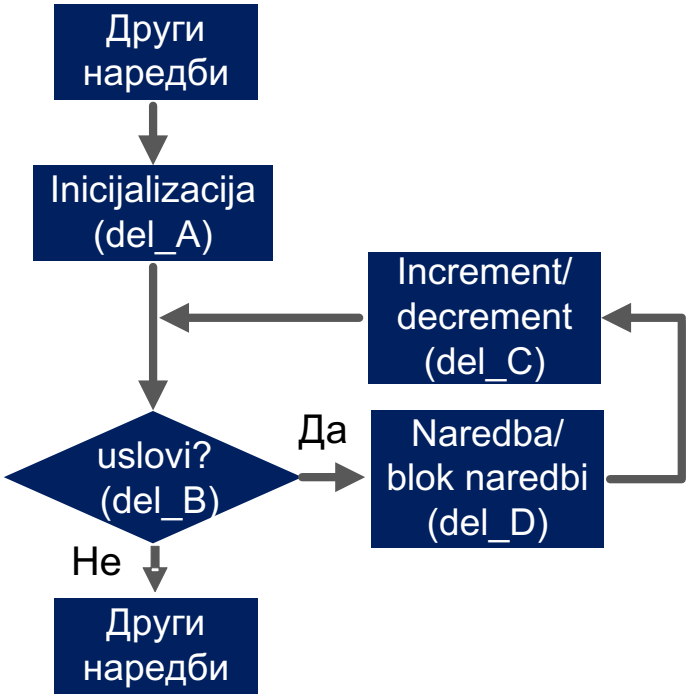
- Наредбите од делот `inicijalizacija` се извршуваат **точно еднаш**, на почетокот - пред влезот во циклусот
- Наредбите од делот `uslovi` се извршуваат **пред почетокот на секој нов циклус**
  - ако резултатот е вредност која се интерпретира како **логичка вистина** се **повторуваат** наредбите од циклусот;
  - **инаку** се **завршува** повторувањето на циклусот и се продолжува со наредбите **зад (по) циклусот**
- Наредбите од делот `inkrementi/dekrementi` се извршуваат **на крајот на секој циклус**
  - по извршувањето на сите наредби од телото на циклусот `blok_naredbi;`
  - потоа се извршуваат наредбите од делот `uslovi` и ако се задоволени, циклусот се повторува.

РЕДОСЛЕД НА ИЗВРШУВАЊЕ НА ДЕЛОВИТЕ ОД FOR НАРЕДБАТА

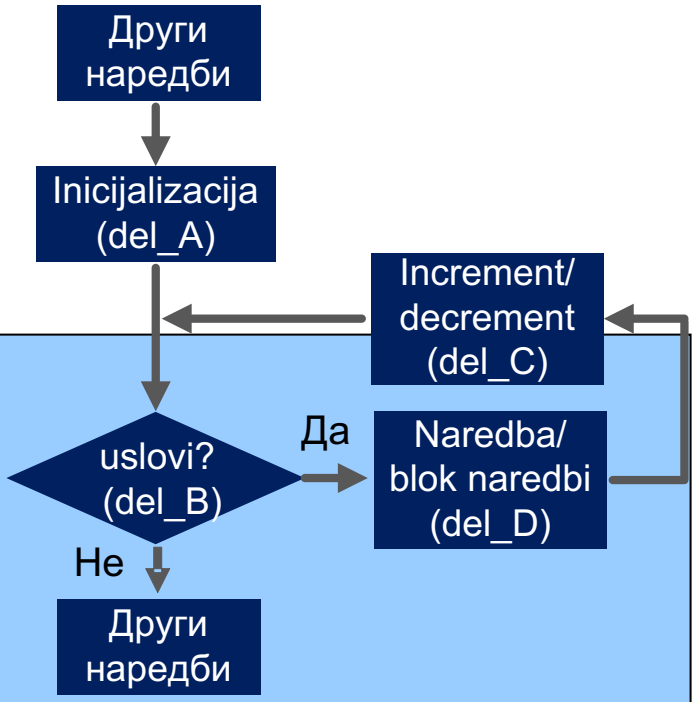
- Редоследот на извршување и интерпретацијата на резултатите е точно одреден.

Илустрацијата со блок дијаграм одговара на следниот програмски код:

```
...
for(del_A; del_B; del_C) {
    del_D;
}
...
```



РЕДОСЛЕД НА ИЗВРШУВАЊЕ НА ДЕЛОВИТЕ ОД FOR НАРЕДБАТА (II)



Пример:

```
#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 zavrsi.");
    return 0;
}
```

```
Ke pocne ciklus...
ABDCBDCBDCB
Ciklusot zavrsi.
```

НАЈЧЕСТА ГРЕШКА ПРИ КОРИСТЕЊЕ НА FOR ЦИКЛУСИ

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;
}
```

- Доколку нема ; крајот од `for` наредбата тогаш наредбата `suma = suma + x * y;` ќе се изврши четири пати и излезот е 50
- Доколку има ; крајот од `for` наредбата тогаш наредбата `suma = suma + x * y;` ќе се изврши само еднаш и излезот е 25

НАПРЕДНО КОРИСТЕЊЕ НА FOR ЦИКЛУСИ

Некои делови на `for` наредбата можат да бидат празни:

```
#include <stdio.h>
int main() {
    int c=-1;
    printf("Vnesuvaj znakovi:\n(Vnesi x za kraj)\n");
    for (  ;c != 'x'; ) {
        c = getchar();
        putchar(c);
    }
    printf("\nKraj na ciklusot!\n");
    return 0;
}
```

Дали ќе се прикаже x?

- Програмата ја печати буквата што сте ја внеле
- Програмата користи јамка со „чувар“
  - „чувар“ е вредноста `'x'`

НАПРЕДНО КОРИСТЕЊЕ НА FOR ЦИКЛУСИ (II)

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;
}
```

- Што работи програмата?

Програмата бара да внесувате броеви се додека не го внесете бројот 6

НАПРЕДНО КОРИСТЕЊЕ НА FOR ЦИКЛУСИ (III)

Пример програма со повеќе бројачи (има два бројачи кои се ажурираат при секој циклус):

```
#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
```

- Што ќе испечати програмата?

ПРИМЕР ЗА ЦИКЛУСИ

Што печати следнава програма?

```
#include <stdio.h>
int main() {
    int j = 0;
    while (j < 3) {
        printf( "Ha "); j++;
    }

    do
    {
        j -= 2; printf( "Hi "); ++j;
    }
    while ( j );

    for(j = 1; j <= 3; j++)
        printf( "Ho ");

    printf("\n");
    return 0;
}
```

```
Ha Ha Ha Hi Hi Hi Ho Ho Ho
```

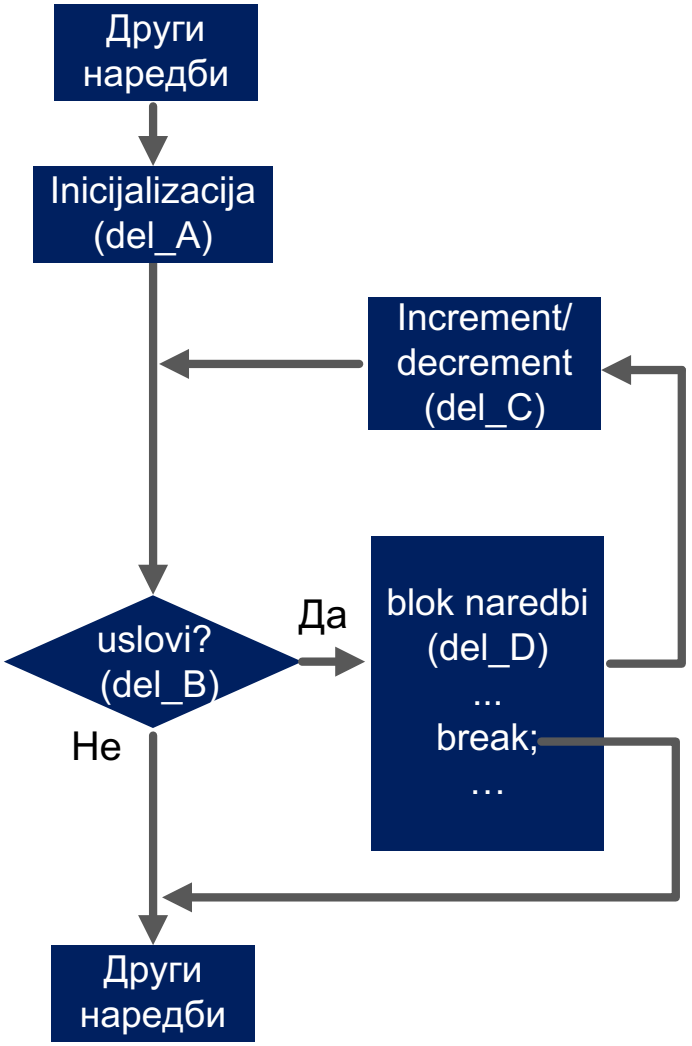
НАРЕДБИ ЗА ИЗЛЕГУВАЊЕ ОД ЦИКЛУС BREAK

- `break`
  - Наредбата `break` овозможува **моментно излегување** од циклус реализиран со `for`, `while`, `do-while` или `switch` **пред условот за напуштање на циклусот да биде исполнет**.
  - Извршувањето продолжува со следната наредба веднаш по `for`, `while`, `do-while` или `switch` структурата
  - Голема примена за оптимизирање на кодови (да се отстрани непотребното извршување на наредби)
- Се користи за
  - Предвремено завршување на јамката
  - Прескокнување на остатокот од `switch` структура (подоцна!!!)

ПРИМЕНА НА BREAK КАЈ FOR ЦИКЛУС

Илустрацијата со блок дијаграм одговара на следниот програмски код:

```
...
for(del_A; del_B; del_C) {
    //del_D;
    …
        break;
    …
}
...
```



- По извршувањето `break` наредбата веднаш потоа се продолжува со наредбите после циклусот
- Во циклусот, наредбата `break` е секогаш наредба во некоја `if` структура

НАРЕДБИ ЗА ИЗЛЕГУВАЊЕ ОД ЦИКЛУС CONTINUE

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

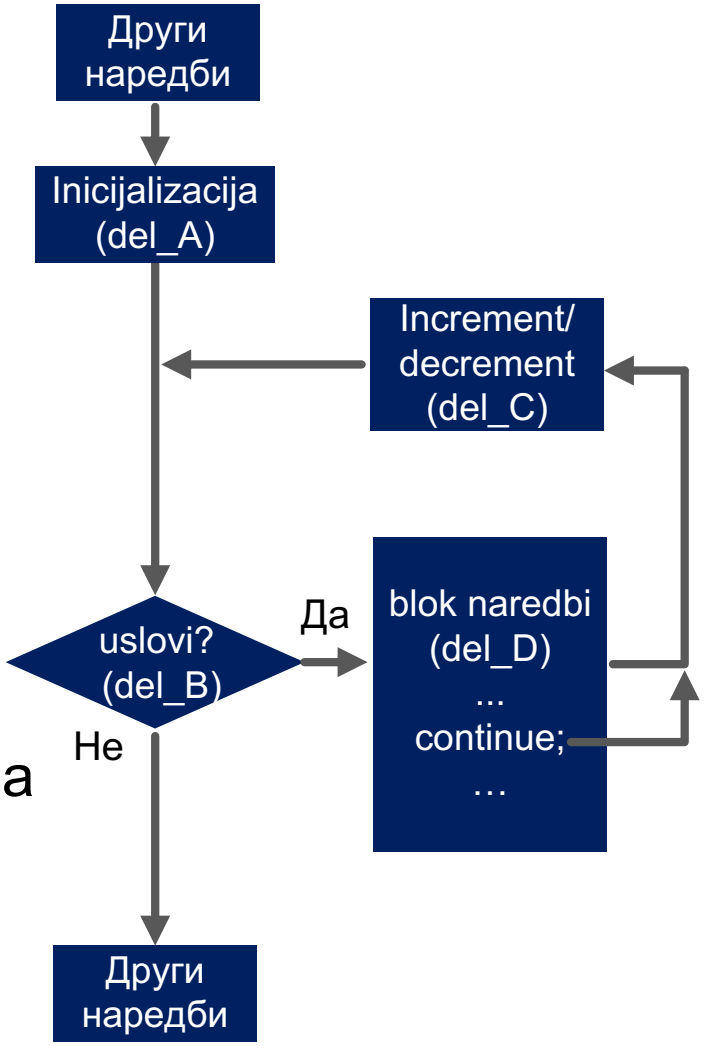
- Kaj `while` и `do/while`
  - По извршување на `continue`, веднаш се извршува проверка на условот од наредбата

- Kaj `for`
  - По извршување на `continue`, се извршува `inkrement/dekrement` изразот, а потоа се проверува условот

ПРИМЕНА НА CONTINUE КАЈ FOR ЦИКЛУС

Илустрацијата со блок дијаграм одговара на следниот програмски код:

```
...
for(del_A; del_B; del_C) {
    //del_D;
    …
        continue;
    …
}
...
```

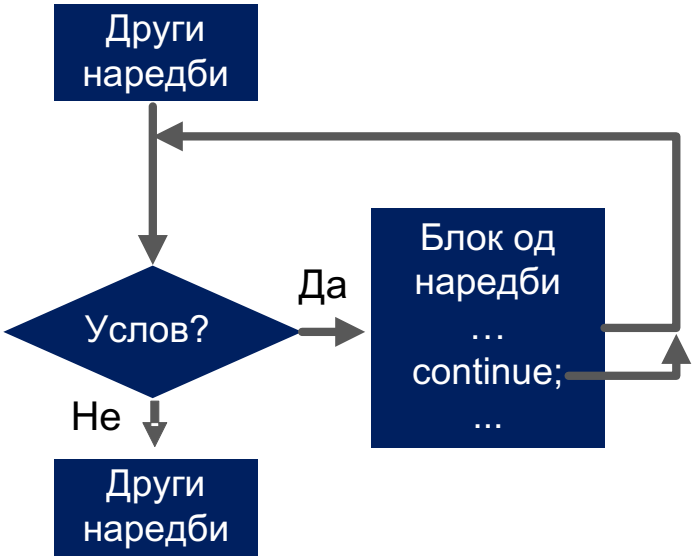


- По извршување на `continue`, се извршува `inkrement/dekrement` изразот, а потоа се проверува условот
- Во циклусот, наредбата `continue` е секогаш наредба во некоја `if` структура

ПРИМЕНА НА CONTINUE КАЈ WHILE ЦИКЛУС

Илустрацијата со блок дијаграм одговара на следниот програмски код:

```
...
while (uslov) {
    //blok_naredbi;
    …
        continue;
    …
}
...
```

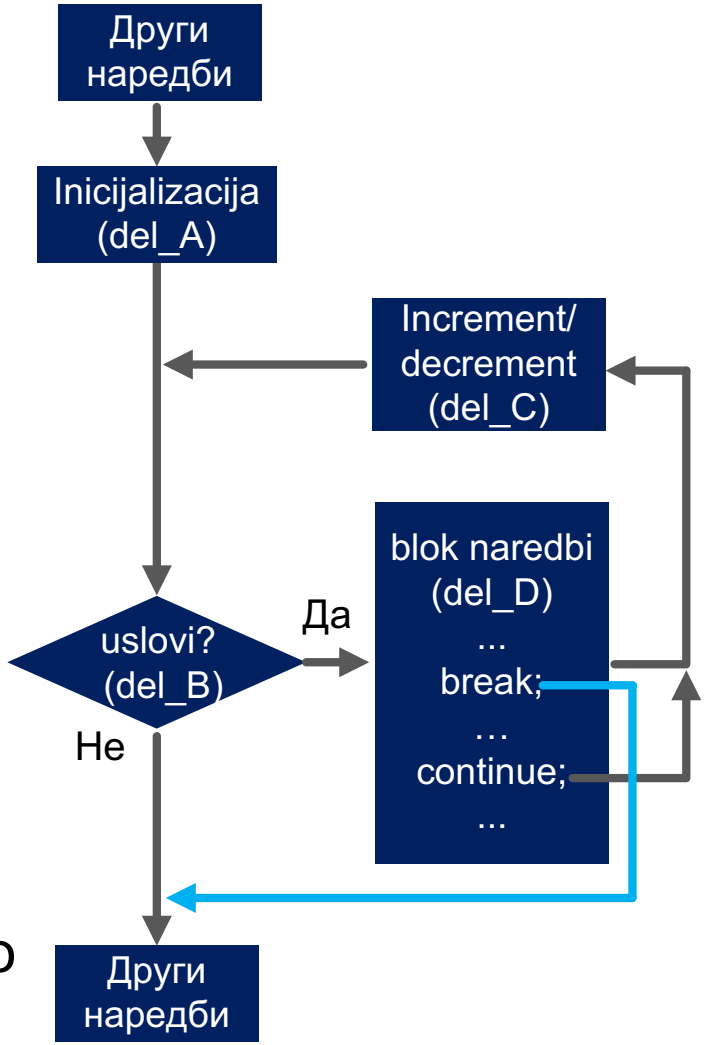


- По извршување на `continue`, веднаш се извршува проверка на условот од наредбата
- Во циклусот, наредбата `continue` е секогаш наредба во некоја `if` структура

ПРИМЕНА НА BREAK И CONTINUE КАЈ FOR ЦИКЛУС

Илустрацијата со блок дијаграм одговара на следниот програмски код:

```
...
for(del_A; del_B; del_C) {
    //del_D;
    …
        break;
    …
        continue;
    …
}
...
```



- Во циклусот, наредбите `continue` и `break` се секогаш дел од наредбите во некоја `if` структура

ПРИМЕР ЗА ПРИМЕНА НА BREAK КАЈ FOR ЦИКЛУС

Пример: Да се напише програма која ќе го пресметува збирот на сите броеви кои може да се запишат како степен од 2 каде степенот е природен број помал од 6…. 2 + 4 + 8 + 16 + 32

```
#include <stdio.h>
int main() {
    int sum = 0, x = 1, b=2;

    for (;;++x) {
        if(x > 5)
            break;
        sum = sum + b; b = b*2;
    }
    printf("Zbirot e %d",sum);
    return 0;
}
```

```
Zbirot e 62
```

- Што ќе испечати програмата?

ПРИМЕРИ ЗА ПРИМЕНА НА CONTINUE

Пример 1: Од тастатура да се внесат непознат број знаци кои ќе се печатат на екран само доколку не се цифри. Да се прекине со извршување кога ќе се внесе знак EOF (ctrl+z за Windows и ctrl+d за Linux)

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

Пример 2:

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

```
5 6 7 8 9
```

Што ќе испечати програмата?

ASCII Code табела

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

ВГНЕЗДЕНИ СТРУКТУРИ ЗА ПОВТОРУВАЊЕ

- Задача: Да се пресмета средна вредност од поените освоени на 3 тестови (кои ќе се внесуваат од тастатура) за 3-ца студенти
- Потребни се два циклуси:
  - надворешен во кој ќе се менува редниот број на студентот и ќе работи се што се однесува на тој студент (ажурирање на променливите за внатрешниот циклус, повикување на внатрешниот циклус и печатење на резултатот)
  - внатрешен што ќе ги врши сите преземања на поените од тестовите и пресметките за студентот чиј реден број се разгледува
- Во секоја итерација на надворешниот циклус, целосно се извршува внатрешниот циклус (пр. кога надворешниот циклус ќе биде кај студент 2, внатрешниот циклус кој ги собира поените ќе направи три итерации (целосно ќе се изврши) за да ги собере сите поени од тестовите кои се однесуваат на студентот 2)

ВГНЕЗДЕНИ СТРУКТУРИ ЗА ПОВТОРУВАЊЕ (II)

- По хоризонтала и со посветла боја е надворешниот циклус кој се извршува 3 пати, а секоја негова итерација се извршува по вертикала а содржи и целосно извршување на внатрешниот циклус кој е со потемна боја

Циклус 1 (stud=1..3)

Student1(stud=1)	Student2(stud=2)	Student3(stud=3)
Циклус 2 (t=1..3)	**Циклус 2 (t=1..3)**	**Циклус 2 (t=1..3)**
poeni za t_1	poeni za t_1	poeni za t_1
poeni za t_2	poeni za t_2	poeni za t_2
poeni za t_3	poeni za t_3	poeni za t_3
$(t_1+t_2+t_3)/3$	$(t_1+t_2+t_3)/3$	$(t_1+t_2+t_3)/3$
Prikazi prosek Student1	Prikazi prosek Student2	Prikazi prosek Student3

ВГНЕЗДЕНИ СТРУКТУРИ ЗА ПОВТОРУВАЊЕ (III)

```
#include <stdio.h>
int main() {
    int stud, t, S, poeni;
    for(stud=1; stud<=3; stud++) { /* nadvoreshen ciklus */
        S=0; /* kontrolna promenliva za vnatreshen ciklus mora
          da se azhurira vo ramkite na nadvoreshniot ciklus */
        for(t=1; t<=3; t++) { // vnatreshen ciklus
            printf("Vnesi poeni za t%d za Student%d\n",t,stud);
            scanf("%d", &poeni);
            S=S+poeni;
        }
        printf("Prosekot za Student%d e %f\n",stud,(float)S/3);
    }
    return 0;
}
```

ПРИМЕР ЗА ВГНЕЗДЕНИ СТРУКТУРИ ЗА ПОВТОРУВАЊЕ

- Задача: Да се состави програма што ќе ги отпечати сите прости броеви помали од даден број.
  - Надворешниот циклус ќе ги разгледува сите броеви помали од дадениот
  - Внатрешниот циклус ќе проверува дали моментално разгледуваниот број од надворешниот циклус е прост број (дали е делив со броевите помали од него) преку проверка на неговата деливост со сите помали броеви од моментално разгледуваниот број (освен со бројот 1).
  - Постојат повеќе верзии за реализација и на надворешниот и на внатрешниот циклус кои се дополнително оптимизирани (се завршуваат во помал број на операции – такви примери ќе се разгледуваат на аудиториски вежби)

```c
#include<stdio.h>
int main() {
        int i,j,prost,n,x = 0;
        printf("Vnesi broj ");
        scanf("%d",&n);
        printf("Prosti broevi pomali od %d se\n",n);
        for(i = 1; i < n; i+=1) { /* nadvoreshen ciklus */
                prost = 1; /* kontrolna prom. za vnatr. ciklus */
                for(j = 2; j < i; j++) /* vnatreshen ciklus */
                        if(i%j == 0) {
                                prost = 0; break;
                        }
                if(prost) { /* ako kontrolnata promenliva
                        ostane 1 ne se nashol delitel */
                        printf("%d ",i);   x++;
                }
        }
        printf("\n Vkupno %d prosti broevi",x);
        return(0);
}
```

Што ако го нема ова break?

Програмирање и алгоритми

СТРУКТУРИ ЗА ИЗБОР И БЕЗУСЛОВЕН СКОК

СТРУКТУРА ЗА ИЗБОР ОД ПОВЕЌЕ МОЖНОСТИ SWITCH

- Корисна кога променлива/израз треба да се тестира за сите можни вредности кои може да ги прими, а притоа е потребно да се извршат различни акции
- Синтакса:

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

  - *izraz* мора да резултира во `int` или `char` тип
  - `default` е опционен и не мора да го има
  - `break;` предизвикува излез од структурата (крај на `switch`)

СТРУКТУРА ЗА ИЗБОР ОД ПОВЕЌЕ МОЖНОСТИ SWITCH (ПРАВИЛА)

- **Не смее** да има **два или повеќе** `case` **изрази со иста вредност**.
- Програмата продолжува со наредбите **зад** `case` наредбата **кој одговара на** вредноста на **пресметаниот израз** од `switch` наредбата.
- Со извршувањето на следните наредби (од следните `case` случаи) се продолжува сè додека не се наиде 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.
- Ако за избраниот `case` нема `break` се продолжува со извршување на наредбите на следниот `case` независно дали одговара

SWITCH (ПРИМЕР)

Програмата овозможува внесување на еден знак од тастатура и доколку станува збор за цифра ја испишува со букви вредноста на цифрата, доколку не станува збор за цифра тогаш се испишува `"ne e cifra"`

```
#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;
}
```

SWITCH (ПРИМЕР 2)

Што ќе отпечати следниот програм?

```
#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
```

SWITCH (ПРИМЕР 3)

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

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

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

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 избегнува користење на оваа наредбата**

КРАЈ