Procedurálne programovanie

2. prednáška

Riadiace Štruktúry

Anna Bou Ezzeddine

acm International Collegiate Programming Contest

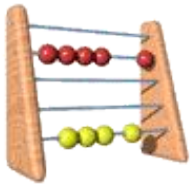
STU FIIT

ACM ICPC je opäť tu!

Lokálne kolo programátorskej súťaže na STU v rámci

CTU Open Contest

27. - 28. 10. 2017



Pošli registračný e-mail na

acm.icpc@fiit.stuba.sk

do stredy 25. 10. 2017 do 18.00 hod.

Viac informácii na:

www.fiit.stuba.sk/acm

ESET GRATEX INTERNATIONAL Nadácia pre rozvoj informatiky

Obsah prednášky

1. Opakovanie
2. riadiace štruktúry
   a) príkazy vetvenia (`?:`, **`if-else`**, **`switch`**)
   b) príkazy cyklov (**`while`**, **`do-while`**, **`for`**)
3. príklady

Cieľ prednášky

- Naučiť sa:
  - vytvárať bloky príkazov
  - zapisovať podmienky
  - definovať a riadiť cykly s podmienkami na začiatku aj na konci cyklu
  - používať vnorené cykly

Ako byť dobrým programátorom?

- Úlohou programátora je **vytvoriť riešenie problému využitím počítačového programu**
  - **Program môže byť opakovane použitý pre rôzne výskyty toho istého problému.**
- Programátor musí:
  1. Porozumieť problému
  2. Vymyslieť a navrhnúť riešenie
  3. Vyjadriť riešenie v programovacom jazyku

- **Najlepší** programátori **PRAVIDELNE** robia chyby.

- **Väčší riešený problém rozdeľte na menšie podproblémy (funkcie). Programujte po malých kúskoch, a každý kúsok si dôsledne overte, či je správny.**

Opakovanie

PROCEDURÁLNE PROGRAMOVANIE

Ako sa vykonáva tento jednoduchý program v pamäti?

```c
#include<stdio.h>
int main()
{
    int a,b,c;
    scanf("%d %d", &a, &b);
    c = a+b;
    printf("a = %d, b = %d, c = %d", a,b,c);
return 0;
}
```

Príklad

Pamäť

???

Premenné a rozsah ich platnosti (bloky)
Kde inicializovať premenné?

- Premenná - previazanie pamäti s nejakým menom
- Deklarácia: `int a, b, c;` -- tri (rôzne) premenné typu int
- Program môže mať rôznu dĺžku a štruktúru...
- **Rozsah platnosti** premenných nám hovorí, **kde v programe** môžeme meno premennej použiť, a **s ktorou pamäťou** je meno premennej previazané
- **Blok** – časť kódu, ktorá je zoskupená spolu, typicky **ohraničená zloženými zátvorkami { }**
- Príklad (rozsah platnosti premenných vyznačený stĺpcami):

a
c

```
int a = 5, c ;
if (...)
{
  int b = 4;
  c = 20;
}
b = b+1;
```

b

CHYBA premenná b na tomto mieste neexistuje!!!

Rozsah platnosti premenných lokálna, globálna premenná

- **Blok** – rozsah platnosti v rámci bloku
- **Funkcia** – rozsah platnosti v rámci volania funkcie (resp. v rámci bloku tela funkcie)
- **Globálny** – rozsah platnosti vo všetkých funkciách
- Čo ak v rozličných rozsahoch platnosti je premenná rovnakého názvu?
  Ak je x v globálnom rozsahu (globálna premenná x) a deklarujeme aj nejakú novú premennú x vo funkcii (lokálna premenná x), tak sa vo funkcii vyhradí nová pamäť pre premennú x v rozsahu platnosti funkcie a v čase vykonávania tejto funkcie do globálneho x nie je možné prostredníctvom mena 'x' zapisovať
- **Názorný príklad ukladania globálnej a lokálnej premennej do pamäte.**

Typová konverzia

- Typ premenných – nutné určiť
- Pri zmene typu premennej počas vykonávania programu je nutná typová konverzia (pretypovanie premennej).
  - Implicitná (samovoľná, automatická)
  - Explicitná (vynútená, požadovaná)

príklad zaokrúhľovanie, alebo odstránenie desatinnej časti reálneho čísla

```c
#include <stdio.h>

int main(void)
{
  double realne;
  int cele;

  printf("Zadaj realne cislo: ");
  scanf("%lf", &realne);
  printf("Vypis cisla bez desatinnej casti:\n");
  printf("1.: %.0lf (format tlace - zaokruhluje)\n", realne);
  printf("2.: %d (explicitna typova konverzia)\n", (int)realne);
  cele = realne;
  printf("3.: %d (implicitna typova konverzia)\n",cele);
return 0;
}
```

Vyskúšajte možnosť použitia knižnice

#include <math.h>

a funkcií

floor(realne); a **ceil(realne);**

round(realne);

Pozor na rozdiel medzi = a ==

- = symbol priradenia
- == symbol porovnania
-

```
int a,b;

a = b;

if(a == b)
{
…
}
```

príklad1: skrátené vyhodnocovanie výrazov

```c
#include <stdio.h>
int main(void)
{
  int i = 20, j = 4;
  if ((i %= j) && ++j == 5)
  printf ("Logicky sucin\n");
  printf("i: %d  j: %d\n", i, j);
return 0;
}
```

Čo vypíše program?

príklad2: skrátené vyhodnocovanie výrazov

```c
#include <stdio.h>
int main(void)
{
  int i = 15, j = 5;

  if ((i %= j)==0 || ++j)
  printf("i = %d  j = %d\n", i, j);
  else
  printf("i = %d  j = %d\n", i+1, j+1);
return 0;
}
```

Čo vypíše program?

príklad3: skrátené vyhodnocovanie výrazov

```c
#include <stdio.h>
int main()
{
   int i = -1, j = 0, k = 1;
   if (i++ && !j || k++)
      j++;
   if (i++ || j && !i || k++)
      j++;
   printf("%d %d %d\n", i, j, k);
return 0;
}
```

Čo vypíše program?

Opakovanie príklad if

program načíta **znak** z klávesnice a
ak je to číslica, vypíše správu

```
#include <stdio.h>
int main()
{
   int c;
   if( c=getchar() >='0'  &&  c <='9')
      printf("cislica");
return 0;
}
```

Kde je chyba?

Opakovanie if else

```c
#include <stdio.h>
int main (void)
{
  char c;
  printf ("Zadaj znak:\n");
  scanf ("%c", &c);

  if ( (c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z') )
   printf ("Abecedny znak\n");
  else if ( c >= '0' && c <= '9' )
   printf ("Cislica\n");
  else
   printf ("Specialny znak\n");
return 0;
}
```

Opakovanie príklad: priestupný rok

program zistí, či rok je priestupný

1. Rok je priestupný, ak je deliteľný 4
2. Rok, ktorý je deliteľný 100 nie je priestupný
3. Rok, ktorý je deliteľný 400 je priestupný
   - 1700, 1800, 1900, 2100 a 2200 nie sú priestupné

Ako by sme to prepísali do jednej podmienky
(jedna **`if-else`** konštrukcia)

Príklad: priestupný rok

```c
#include<stdio.h>
int main()
{
   int rok;

   printf("Zadajte rok: ");
   scanf("%d", &rok);

    if(((rok % 4 == 0)&&(rok % 100 != 0))||(rok % 400 == 0))
      printf("%d je priestupny rok", rok);
    else
      printf("%d nie je priestupny rok", rok);

return 0;
}
```

Riadiace štruktúry

Viacnásobné vetvenie

Cykly

Mnohonásobné vetvenie

```
if (c == 'a')
    ...
else if (c == 'b')
    ...
else if (c == 'c')
    ...
else if (c == 'd')
    ...
else
    ...
```

jednoduchšie: príkazom **`switch`**

Príkaz **`switch`**

- výraz, podľa ktorého sa rozhoduje, musí byť typu **`int`**
- každá vetva **by mala byť** ukončená príkazom **`break`**
- v každej vetve môže byť viac príkazov, ktoré nie je nutné uzatvárať do zátvoriek
- vetva **`default`** - vykonáva sa, keď žiadna iná vetva nie je splnená

```
switch (vyraz) {
   case hodnota_1 : prikaz_1; break;
   ...
   case hodnota_n : prikaz_n; break;
   default : prikaz_def; break;
}
```

Príklad

Jednoduchá kalkulačka

Po vybraní typu operácie budú zadané hodnoty operandov a vypočítaný výsledok.
Použitie príkazu switch

```c
#include <stdio.h>
int main()
{
    int operacia, n1, n2, vysledok;

    printf("Aku operaciu chcete vykonat?\n");
    printf("stlac 1 na scitanie\n");
    printf("stlac 2 na odcitanie\n");
    printf("stlac 3 na nasobenie\n");
    printf("stlac 4 na delenie\n");
    scanf("%d", &operacia);
    printf("Zadaj prve cislo\n");
    scanf("%d", &n1);
    printf("Zadaj druhe cislo\n");
    scanf("%d", &n2);
```

Pokračovanie ->

```
switch(operacia)
    {
        case 1: vysledok  = n1+n2;
        printf("Vysledok scitania je %d\n", vysledok);
        break;
        case 2: vysledok  = n1-n2;
        printf("Vysledok odcitania je %d\n", vysledok);
        break;
        case 3: vysledok  = n1*n2;
        printf("Vysledok nasobenia je %d\n", vysledok);
        break;
        case 4: vysledok  = n1/n2;
        printf("Vysledok delenia je %d\n", vysledok);
        break;
        default:printf("Chyba");
    }

return 0;

}
```

Iteračné príkazy - cykly

- umožňujú opakovať vykonávanie príkazu alebo bloku príkazov
- tri príkazy: **`while`**, **`for`**, **`do-while`**
- vo všetkých typoch cyklov je možné použiť príkazy na zmenu "normálneho" behu cyklu:

```
break
```

ukončuje cyklus

```
continue
```

prechádza na ďalšiu iteráciu

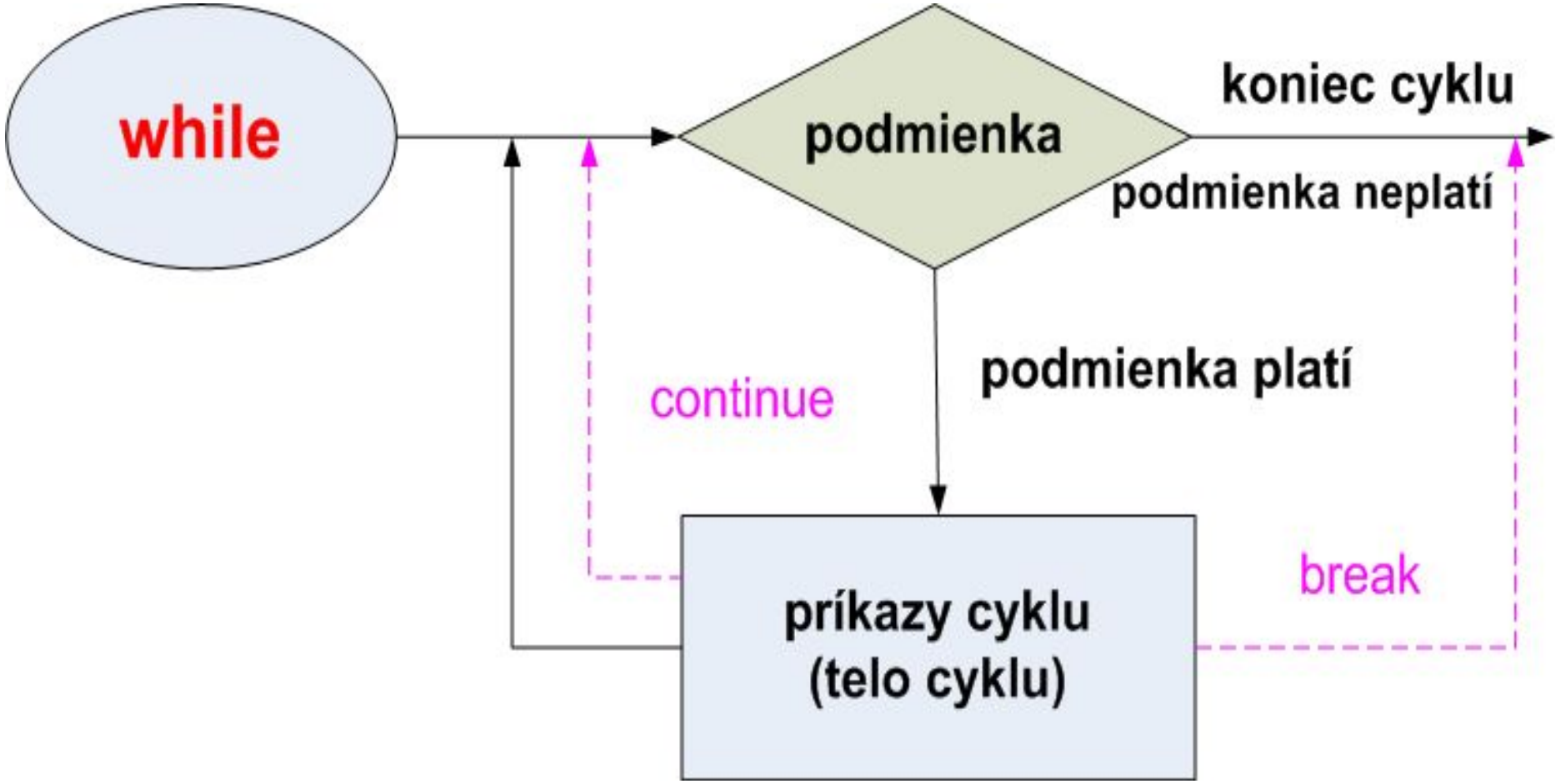
Príkaz **`while`**

- cyklus iteruje pokiaľ platí **`podmienka:`**

```
while (podmienka)
    prikaz;
```

- testuje podmienku **pred** prechodom cyklu
  - cyklus teda nemusí prebehnúť ani raz
- používame ho, ak ukončovacia podmienka závisí na nejakom príkaze v tele cyklu
  - ak by bola podmienka splnená stále cyklus by bol nekonečný , napr. **`while (1)`**

cyklus **while**



Príklad NSD, použitie cyklu while

```c
#include <stdio.h>
int main()
{
    int u, v, pom;
    printf ("Zadaj dve kladne cele cisla\n");
    scanf ("%d %d", &u, &v);

    while ( v != 0 ) {
    pom = u % v;
    u = v;
    v = pom;
    }

    printf ("NSD zadanych cisel je: %d\n", u);
return 0;
}
```

Príklad: reverzné číslo, použitie cyklu while

```c
#include <stdio.h>
int main (void)
{
    int cislo, reverz_cislo;

    printf ("Zadaj cele cislo: ");
    scanf ("%d", &cislo);

    while ( cislo != 0 ) {
        reverz_cislo = cislo % 10;
        printf ("%d", reverz_cislo);
        cislo = cislo / 10;
    }
    printf ("\n");
return 0;
}
```

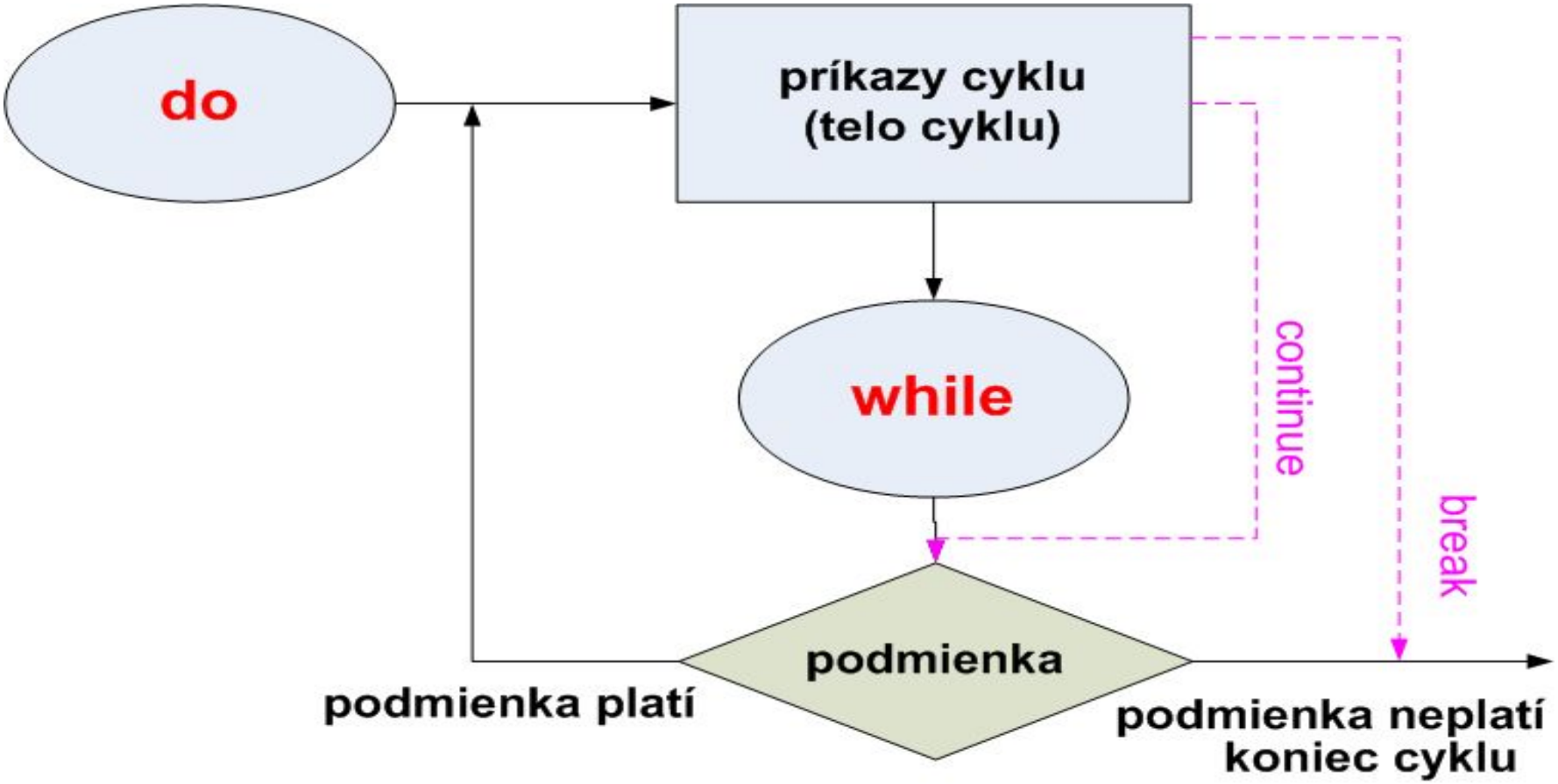
Príkaz **do-while**

- testuje podmienku po prechode cyklu
  - cyklus sa vykoná aspoň raz

```
do {
   prikazy;
}while (podmienka)
```

- program opúšťa cyklus pri nesplnenej podmienke

cyklus **do while**

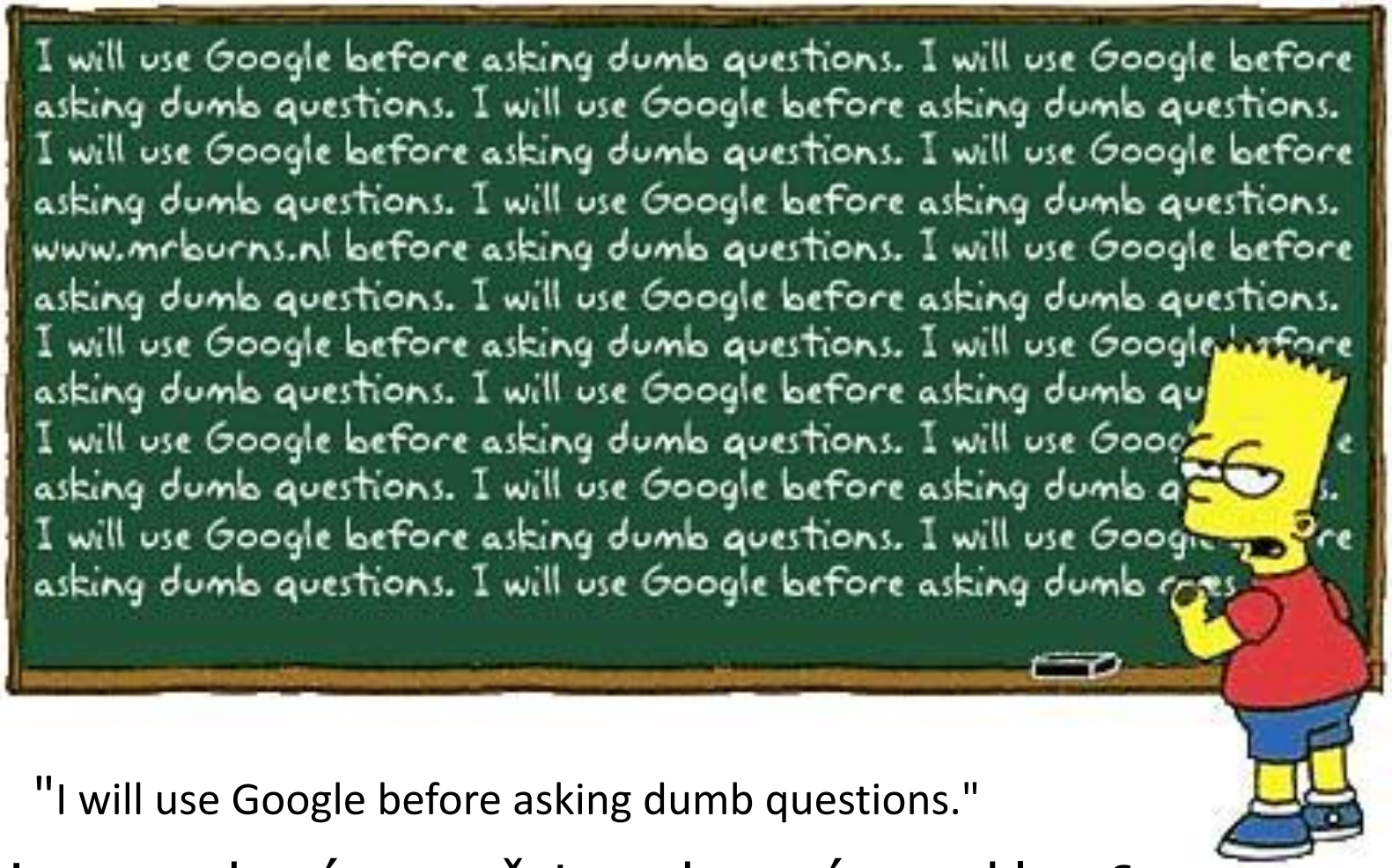


Príklad: reverzné číslo, použitie cyklu do while

Výpis reverzného čísla

```c
#include <stdio.h>
int main ()
{
    int cislo, reverz_cislo;
    printf ("Zadaj cele cislo:");
    scanf ("%d", &cislo);
    do {
        reverz_cislo = cislo % 10;
        printf ("%d", reverz_cislo);
        cislo = cislo / 10;
    }
    while ( cislo != 0 );
    printf ("\n");
return 0;
}
```

Napíš 100x ...



"I will use Google before asking dumb questions."

Ak je vopred známy počet opakovaní → cyklus **for**

Príkaz **for**

- používa sa, keď dopredu vieme počet prechodov cyklom

```
for (vyraz_start; vyraz_stop; vyraz_iter)
   prikaz;
```

Vyhodnotí sa **vyraz_start**,zistí sa pravdivosť **vyraz_stop**, ak je pravdivý vykonajú sa príkazy v tele cyklu, potom ~~sa zvýši iterácia a nasleduje~~ ďalší cyklus. Cyklus končí ak výraz **vyraz_stop** **nie je pravdivý**

napíš 100x "**I will not cut corners**" - vždy do nového riadku, každý riadok začni číslom riadku

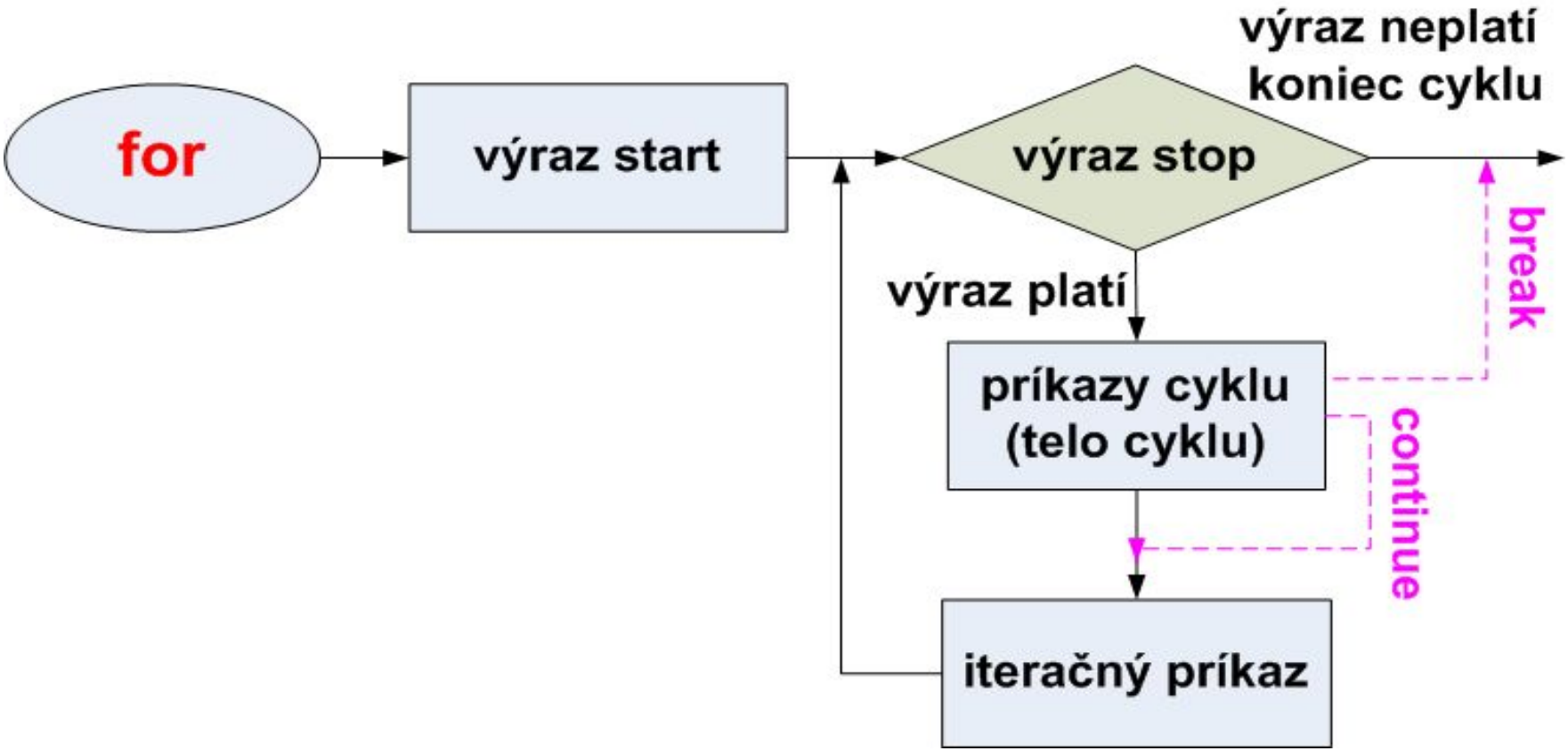
```
for (i = 1; i <= 100; i++)
   printf("%d: I will not cut corners. \n", i);
```

Príkaz **for**

```
for (vyraz_start; vyraz_stop; vyraz_iter)
   prikaz;
```

- výrazy **vyraz_start**, **vyraz_stop**, **vyraz_iter** nemusia spolu súvisieť a nemusia byť vôbec uvedené- v každom prípade treba uviesť bodkočiarku.
- **for(;;)**nekonečný cyklus
- priebeh for-cyklu:
  1. na začiatku sa vyhodnotí **vyraz_start**
  2. otestuje sa, či je **vyraz_stop** pravdivý,ak je nepravdivý skončí
  3. ak je pravdivý, vykoná sa **prikaz** a vykoná sa **vyraz_iter**
  4. návrat na začiatok cyklu (ďalšia iterácia)
- dajú sa použiť **break** a **continue**

cyklus **for**


for
výraz start
výraz stop
výraz neplatí
koniec cyklu
break
výraz platí
príkazy cyklu
(telo cyklu)
continue
iteračný príkaz

Príklad cyklus for

Výpis čísel od 1 po n

Kde je chyba?

```c
#include <stdio.h>

int main()
{
   int i, n;
   scanf("%d", &n );

   for (i=1; i<=n; i++)
         printf("%d\n", i++);

return 0;
}
```

Príkaz `for`

```
for (vyraz_start; vyraz_stop; vyraz_iter)
    prikaz;
```

- dá sa prepísať ako while cyklus:

```
vyraz_start;
while (vyraz_stop) {
    prikaz;
    vyraz_iter;
}
```

Príklad cyklus **while**

Výpis čísel od 1 po n

```c
#include <stdio.h>

int main()
{
   int i = 1, n;
   scanf("%d", &n );

   while(i<=n){
       printf("%d\n", i);
       i++;
   }
return 0;
}
```

Príklad cyklus **do while**

Výpis čísel od 1 po n

```c
#include <stdio.h>

int main()
{
   int i = 1, n;
   scanf("%d", &n );
   if(n >= 1)
         do {
            printf("%d\n", i++);
         } while(i<=n);
return 0;
}
```

Odporúčania

- mať len jednu riadiacu premennú
- inicializácia v inicializačnej časti
- príkaz **`continue`** je vhodné nahradiť **`if-else`** konštrukciou
- príkaz **`break`** - len v nutných prípadoch, najlepšie maximálne na jednom mieste
- cykly **`while`** a **`for`** sú prehľadnejšie ako **`do-while`**, preto ich uprednostňujte

Príklad výpis písmen - opakovane

```c
#include <stdio.h>
int main() {
    int c1, c2;
    int i, n;

    printf("Zadajte velke pismeno: ");
    c2 = getchar();
    if(c2 >= 'A' &&   c2 <= 'Z') {
       printf("Kolkokrat vypisat A - %c? ", c2);
       scanf("%d", &n);

       for(i=1; i<=n; i++) {
         for(c1='A'; c1<=c2; c1++)
             putchar(c1);
         putchar('\n');
       }

    }
return 0;
}
```

Výpis písmen od A po zadané písmeno – zvolený počet krát.

pre c2=H, n=6

```
ABCDEFGH
ABCDEFGH
ABCDEFGH
ABCDEFGH
ABCDEFGH
ABCDEFGH
```

Príklad: doplňte chýbajúce príkazy

Výpis prvých 1, 2, 3, ...n písmen po zadané n v uvedenom tvare:

```
#include <stdio.h>

int main() {
   int i, j, n;
   printf ("Zadaj pocet pismen<ako 27:");
   scanf("%d", &n);

   for(i=1; i<=n; i++) {
      for(j=0; j<i; j++)
         putchar('A'+j);
      putchar('\n');
   }
return 0;
}
```

pre n: 5

A
AB
ABC
ABCD
ABCDE

Vypíšte Floydov trojuholník

rozmer=20;

```c
#include <stdio.h>
int main( )
{
     int i, j, k = 1, rozmer;
    printf("zadaj rozmer Floydovho trojuholnika\n");
    scanf("%d", &rozmer);

          for(i = 1; k <= rozmer; ++i)
          {
            for(j = 1; j <= i; ++j)
           printf("%d  ", k++);
           printf("\n");
     }
return 0;
}
```

```
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
```

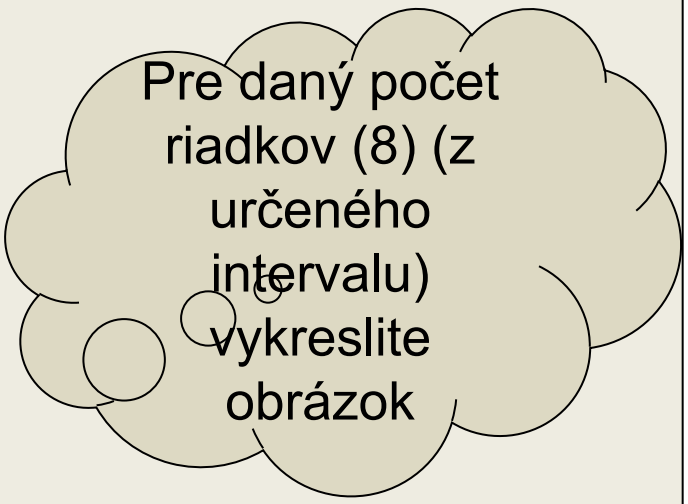
Príklad: hviezdičkovanie trojuholníka

```c
#include <stdio.h>
int main() {
    int i, j, n, r;

    scanf("%d", &n);
    for(i=1; i<=n; i++) {
        for(j=1; j<=n; j++)
            if(i < j)
                putchar('*');
            else
                printf("%d", i%10);
        putchar('\n');
    }
return 0;
}
```

Pre daný počet riadkov (11) vykreslite obrázok

```
1**********
22*********
333********
4444*******
55555******
666666*****
7777777****
88888888***
999999999**
0000000000*
11111111111
```

```c
#include <stdio.h>

int main() {
    int i, j, n;

    scanf("%d", &n);
    if(n < 1 || n > 15) return 0;
    for(i=1; i<=n; i++) {
    for(j=1; j<=2*n-1; j++) {
            if(j <= n-i || j >= n+i)
                putchar('*');
            else
                printf("%d", i%10);
        }
        putchar('\n');
    }
return 0;
}
```




```
*******1*******
******222******
*****33333*****
****4444444****
***555555555***
**6666666666**
*7777777777*
888888888888888
```

```c
#include <stdio.h>

int main() {
   int i, j, n, cislo;

   scanf("%d", &n);
   if(n < 1 || n > 15) return 0;
   for(i=1; i<=n; i++) {
      cislo = i;
      for(j=1; j<=2*n-1; j++)
         if(j <= n-i || j >= n+i)
            putchar('*');
         else {
            printf("%d",cislo%10);
            cislo++;
         }
      putchar('\n');
   }
return 0;
}
```

Pre daný počet riadkov (7) vykreslite obrázok

```
******1******
*****234*****
****34567****
***4567890***
**567890123**
*67890123456*
7890123456789
```

```c
#include <stdio.h>

int main() {
   int i, j, n, cislo;

   scanf("%d", &n);
   if(n < 1 || n > 15) return 0;
   for(i=1; i<=n; i++) {
      cislo = i;

      for(j=1; j<=2*n-1; j++)
         if(j <= n-i || j >= n+i)
            putchar('*');
         else {
            printf("%d", cislo%10);
            if(j < n/2+1)
               cislo++;
            else
               cislo--;
         }
      putchar('\n');
   }
   return 0;
}
```

Pre daný počet riadkov (8) vykreslite obrázok

```
*******1*******
******232******
*****34543*****
****4567654****
***567898765***
**67890109876**
*7890123210987*
890123454321098
```

Opakovanie

ÚLOHY SI NAPROGRAMUJTE,
ZVÝŠENÚ POZORNOSŤ VENUJTE FAREBNE VYZNAČENÝM ČASTIAM PROGRAMU

príklad: doplňte chýbajúcu časť programu

```
#include <stdio.h>

void main() {
  int i, j, n, sucet;
  printf("Zadajte n: ");
  scanf("%d", &n);

  for (i=1; i<=n; i++) {
    sucet = 0;
    for (j=1; j<=i; j++)
      sucet += j;
    printf("1 - %2d: %2d\n", i, sucet);
  }
}
```

program vypíše súčty

1 + . . . + i

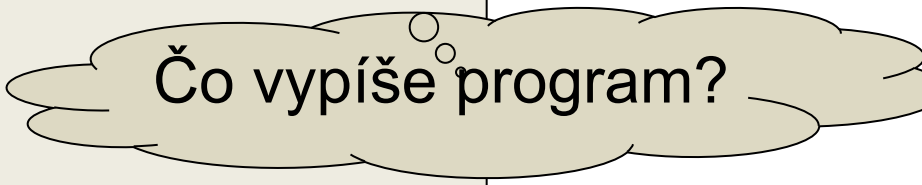
pre všetky i od + do n

počet prechodov cyklom sa zvyšuje

```
pre n: 7

1 -  1:  1
1 -  2:  3
1 -  3:  6
1 -  4: 10
1 -  5: 15
1 -  6: 21
1 -  7: 28
```

Príklad: **break a continue**

```
#include <stdio h>

int main() {
   int i;

   for(i=5; i<=10; i=i+1) {
      if(i == 8)
         break;
      printf("prvy for - i: %d\n", i);
   }
   for(i=5; i<=10; i=i+1) {
      if(i == 8)
         continue;
      printf("druhy for - i: %d\n", i);
   }
return 0;
}
```




```
prvy for - i: 5
prvy for - i: 6
prvy for - i: 7
druhy for - i: 5
druhy for - i: 6
druhy for - i: 7
druhy for - i: 9
druhy for - i: 10
```

Príklad: trojuholník

```
#include <stdio.h>

int main() {
   int i, j, n;

   scanf("%d", &n);
   for(i=1; i<=n; i++) {
      for(j=1; j<=n; j++)
      if(i>=j)
         putchar('*');
      else putchar(' ');

      putchar('\n');
   }
   return 0;
}
```

Program vykrelí trojuholník
z hviezdičiek a medzier

Pre n=5:

```
*
**
***
****
*****
```

Príklad: trojuholník – pridaný `for` (1)

```
#include <stdio.h>

int main() {
   int i, j, n, k;

   scanf("%d", &n);

   for(k=1; k<=2; k++)
      for(i=1; i<=n; i++) {

         for(j=1; j<=n; j++)

            if(i>=j)
       putchar('*');
            else putchar(' ');
         putchar('\n');
      }
   return 0;
}
```

Čo urobí pridanie for-cyklu?

Pre n=5:

```
*
**
***
****
*****
*
**
***
****
*****
```

Príklad: trojuholník – pridaný `for` (2)

```
#include <stdio.h>

int main() {
   int i, j, n, k;

   scanf("%d", &n);


   for(i=1; i<=n; i++) {
      for(k=1; k<=2; k++)
         for(j=1; j<=n; j++)

            if(i>=j) putchar('*');
            else putchar(' ');
         putchar('\n');
      }
   return 0;
}
```

Čo urobí pridanie for-cyklu?

Pre n=5:

```
*     *
**    **
***   ***
****  ****
**********
```

Príklad: trojuholník – pridaný `for` (3)

```
#include <stdio.h>

int main() {
   int i, j, n, k;

   scanf("%d", &n);


   for(i=1; i<=n; i++) {

         for(j=1; j<=n; j++)
            for(k=1; k<=2; k++)
               if(i>=j) putchar('*');
               else putchar(' ');
         putchar('\n');
      }
   return 0;
}
```

Čo urobí pridanie for-cyklu?

Pre n=5:

```
**
****
******
********
**********
```

príklad

```
#include <stdio.h>

int main()
{
    int i, dlzka;
    printf("Zadajte dlzku: ");
    scanf("%d", &dlzka);

    for (i = 1; i <= dlzka; i++)
        if (i % 2)
            putchar('-');
        else
            putchar('*');
return 0;
}
```

do riadku nakreslí striedavo na každú druhú pozíciu hviezdičku

dlzka: 8

-*-*-*-*

príklad

pomocou hviezdičiek nakreslí kríž

```
#include <stdio.h>

int main()
{
    int dlzka, i, j;
    printf("Zadajte dlzku ramena: ");
    scanf("%d", &dlzka);

    for (i = 1; i <= dlzka * 2 + 1; i++) {
        for (j = 1; j <= dlzka * 2 + 1; j++)
            if (j == dlzka+1 || i == dlzka+1)
                putchar('*');
            else
                putchar('-');
        putchar('\n');
    }
return 0;
}
```

dlzka: 3

```
---*---
---*---
---*---
*******
---*---
---*---
---*---
```

príklad

```
#include <stdio.h>

int main()
{
  int r, i, j;
  printf("Zadajte rozmer: ");
  scanf("%d", &r);

  for (i=1; i<=r; i++) {
    for (j=1; j<=r; j++)
      if ((i % 2 == 0 && j % 2 == 1) ||
          (i % 2 == 1 && j % 2 == 0))
        putchar('*');
      else
        putchar(' ');
   putchar('\n');
  }
return 0;
}
```

pomocou hviezdičiek nakreslí uvedený vzor

```
r: 10

 * * * * *
* * * * *
 * * * * *
* * * * *
 * * * * *
* * * * *
 * * * * *
* * * * *
 * * * * *
* * * * *
```

príklad

pomocou hviezdičiek
nakreslí uvedený vzor

```
...
for (i=1; i<=r; i++) {
  for (j=1; j<=r; j++)
  if (i % 2 == 1 && (j % 6 == 1 || j % 6 == 2) ||
      i % 2 == 0 && j % 6 != 1 && j % 6 != 2)
    putchar('*');
  else
    putchar('-');
  putchar('\n');
}
...
```

len zátvorky, ktoré
musia byť

r: 14

```
**----**----**
--****--****--
**----**----**
--****--****--
**----**----**
--****--****--
```

```c
#include <stdio.h>

int main() {
  int i, j, r;

  printf("Zadajte rozmer: ");
  scanf("%d", &r);

  for (i=1; i<=r; i++) {
    for (j=1; j<=r; j++)
      if ((i > 1) && (i < r) &&
          (j > 1) && (j < r))
        putchar('*');
      else
        putchar('-');
    putchar('\n');
  }
return 0;
}
```

pomocou hviezdičiek
nakreslí štvorec v ráme

vnútorné zátvorky
nemusia byť

r: 4

```
----
-**-
-**-
----
```

príklad

```
...
int i, j, k, n, r;
...
for (k=1; k<=n; k++) {
  for (i=1; i<=r; i++) {
    for (j=1; j<=r; j++)
      if ((i > 1) && (i < r) &&
          (j > 1) && (j < r))
        putchar('*');
      else
        putchar('-');
    putchar('\n');
  }
}
```

pomocou hviezdičiek nakreslí štvorce v ráme pod seba

n-krát za sebou zopakujeme kreslenie štvorca

r: 4, n:2

```
----
-**-
-**-
----    k:1
----
-**-
-**-
----    k:2
```

príklad

```
...
int i, j, k, n, r;
...
for (i=1; i<=r; i++) {
  for (k=1; k<=n; k++) {
    for (j=1; j<=r; j++)
      if((i > 1) && (i < r) &&
         (j > 1) && (j < r))
        putchar('*');
      else
        putchar('-');
  }
  putchar('\n');
}
```

pomocou hviezdičiek nakreslí štvorce vedľa seba

n-krát zopakujeme každý riadok

pre **r**: 4, n:2

```
----|----
-**-|-**-
-**-|-**-
----|----
```

Pevné korene na FIIT

NEFAJČIŤ !!!