Algoritmizácia a programovanie
3. prednáška

Ján Grman

Obsah

1. opakovanie
2. príkazy cyklov (**`while, do-while, for`**)
3. Mnohonásobné vetvenie (**`switch`**)
4. príklady

Príklad: operátory

Čo vypíše program? Aké budú hodnoty premenných?

```
#include <stdio.h>

int main() {
   int a=2, b=3, c=4, d=5, e=6;

   printf("%d\n", e / --a * b++ / c++);
   printf("a: %d, b: %d, c: %d, d: %d, e: %d\n", a, b, c, d, e);
   a %= b = d = 1 + e / 2;
   printf("a: %d, b: %d, c: %d, d: %d, e: %d\n", a, b, c, d, e);

   return 0;
}
```

4
a: 1, b: 4, c: 5, d: 5, e: 6
a: 1, b: 4, c: 5, d: 4, e: 6

Príklad: ternárny operátor

program načíta 2 reálne čísla a pomocou ternárneho operátora vypíše ich maximum

```
#include <stdio.h>

int main() {
    float x, y;

    printf("Zadajte 2 realne cisla: ");
    scanf("%f %f", &x, &y);
    printf("%.2f\n", (x > y) ? x : y );
    return 0;
}
```

kde je chyba?

načítavanie na adresu premennej: **&**

Príklad: if

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

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

Kde je chyba?

Príklad: priestupný rok

program zistí, či rok je priestupný

Rok je priestupný:

1. Rok je priestupný, ak je deliteľný 400 (1600, 2000)
2. Ak rok nie je deliteľný 400 ani 100, ale je deliteľný 4, tak je tiež priestupný (napr. 2004, 2008, 1012)

```
IF rok modulo 400 je 0
 THEN priestupny
ELSE IF rok modulo 100 je 0
 THEN nie_je_priestupny
ELSE IF rok modulo 4 je 0
 THEN priestupny
ELSE nie_je_priestupny
```

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

Príklad: priestupný rok

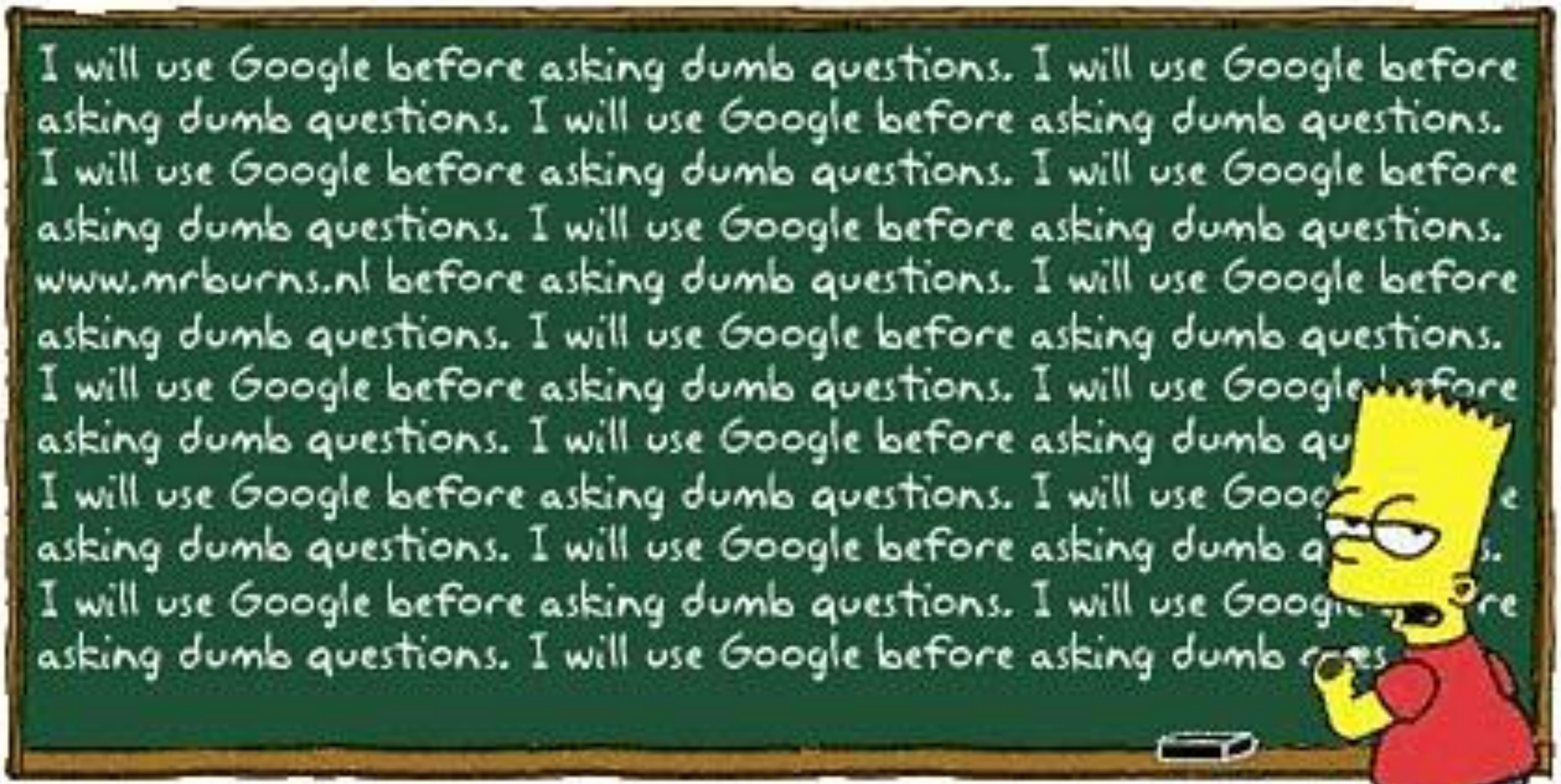
```
#include<stdio.h>
int main(){
   int rok;

   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;
}
```

Napíš 100x ...



"I will use Google before asking dumb questions."

→ cykly

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 (ukončuje najvnútornejšiu slučku a opúšťa cyklus)

`continue` — skáče na koniec najvnútornejšej slučky s tým si vynúti ďalšiu iteráciu

Príkaz `while`

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

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

- testuje podmienku **pred** prechodom cyklu
  - cyklus teda nemusí prebehnúť ani raz
- používame ho, keď ukončovacia podmienka závisí na nejakom príkaze v tele cyklu
  - ak nie, podmienka by bola splnená stále a cyklus by bol nekonečný

Príkaz `while`: príklad

program číta znaky z klávesnice, opisuje ich na obrazovku, medzery si nevšíma a skončí po prečítaní znaku *

```
#include <stdio.h>

int main()
{
    int c;

    while ((c = getchar()) != '*') {
        if (c != ' ')
            putchar(c);
    }
    return 0;
}
```

načítanie znaku - musí byť uzátvorkované, lebo != má väčšiu prioritu ako =

Príkaz `while`: príklad s použitím `break` a `continue`

```
#include <stdio.h>

int main()
{
    int c;

    while (1){
        if ((c = getchar()) == ' ')
            continue;
        if (c == '*')
            break;
        putchar(c);
    }
    return 0;
}
```

program číta znaky z klávesnice, opisuje ich na obrazovku, medzier si nevšíma a skončí po prečítaní znaku *

Príkaz `while`

- telo cyklu môže byť aj prázdne, napr. na vynechanie medzier na vstupe:

```
while (getchar() == ' ')
    ;
```

- alebo preskočí všetky biele znaky na vstupe:

```
while ((c = getchar()) == ' ' || c == '\t' ||
        c == '\n')
    ;
```

Príkaz `do-while`

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

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

- program opúšťa cyklus pri **nesplnenej** podmienke

Príkaz `do-while`: príklad

program číta znaky z klávesnice, opisuje ich na obrazovku, medzier si nevšíma a skončí po prečítaní znaku *, na konci vypíše *

```
#include <stdio.h>

int main()
{
    int c;

    do {
        if ((c = getchar()) != ' ')
            putchar(c);
    } while (c != '*')
    return 0;
}
```

Príkaz `for`

- používa sa, keď dopredu vieme počet prechodov cyklom

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

keď nesplnený **`vyraz_stop`** - končí cyklus

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
- priebeh for-cyklu:
  1. na začiatku sa vyhodnotí **`vyraz_start`**
  2. otestuje sa, či je **`vyraz_stop`** pravdivý, inak skonči
  3. ak áno, vykoná sa **`prikaz`** a vykoná sa **`vyraz_iter`**
  4. na začiatok cyklu (2)
- dajú sa použiť **`break`** a **`continue`**

Príkaz `for`

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

- dá sa prepísať ako while cyklus:

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

Príkaz `for`: príklady

```
int i = 1;
```

všetky 3 príklady predokladajú definíciu, vypisujú čísla od 1 do 10

```
for (i = 1; i <= 10; i++)
   printf("%d ", i);
```

klasické a odporučené použitie

```
for ( ; i <= 10; i++)
   printf("%d ", i);
```

využitie inicializácie v definícii - nevhodné, lebo nie je všetko spolu

```
for ( ; i <= 10; )
   printf("%d ", i++);
```

riadiaca premenná je menená v tele cyklu - nevhodné

Príkaz `for`: príklady

```
for ( ; i <= 10; printf(%d ", i), i++)
   ;
```

využitie operátora čiarka (,) - časté, nie úplne vhodné

```
int i, sum;
for (i = 1, sum = 0; i <= 10; sum += i, i++)
   printf( d ", i);
```

použitie operátora čiarka v inicializácii - vhodné, pri výpočte - nevhodné

Príkaz `for`: príklady

```
int i, sucin;
for (i = 3, sucin = 1; i <= 9; i += 2)
    sucin *= i;
```

cyklus môže meniť riadiacu štruktúru ľubovoľným spôsobom (nielen `i++`)

Odporúčania

- mať len jednu riadiacu premennú
- riadiaca premenná má byť ovplyvňovaná len v riadiacej časti cyklu, nie v jeho tele
- inicializácia v inicializačnej časti
- ak má cyklus (nie len **`for`**) prázdne telo, bodkočiarku dať na nový riadok
- príklaz **`continue`** je vhodné nahradiť **`if-else`** konštrukciou
- príkaz **`break`** - len v najnutnejších prípadoch, najlepšie maximálne na jednom mieste
- cykly **`while`** a **`for`** sú prehľadnejšie ako **`do-while`**, preto ich uprednostňujte

Príklad: výpis čísel

```
#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;
}
```

Výpis čísel od 1 po n

do-while vždy vykoná prvý beh cykom. **if** zabezpečí, že keď n<1, cyklus sa nevykoná ani raz

Príklad: počet deliteľov

Výpis počtu deliteľov zadaného čísla zo zadaného intervalu.

```
#include <stdio.h>

int main() {
   int i, int1, int2, del, pocet=0;

   printf("Zadajte interval a delitel: ");
   scanf("%d %d %d", &int1, &int2, &del);
   for(i=int1; i<=int2; i++)
      if (!(i % del))
         pocet++;
   printf("V <%d, %d> je %d delitelov cisla %d.\n",
      int1, int2, pocet, del);
   return 0;
}
```

Príklad: výpis písmen - opakovane

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

```
#include <stdio.h>

int main() {
   char 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;
}
```

Príklad: výpis 1, ..., n písmen

Výpis prvých 1, 2, 3, ...n písmen po zadané n.

```
#include <stdio.h>

int main() {
   int i, j, n;
   printf("Zadajte pocet: ");
   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

Príklad: Súčty čísel od 1 do i

```
#include <stdio h>

int 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);
  }
  return 0;
}
```

program vypíše súčty
1 + ... + i
pre všetky i od 1 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: Hviezdičkovanie 1

```
#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

pre **dlzka**: 8

```
* * * *
```

Príklad: Hviezdičkovanie 2

```
#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;
}
```

pomocou hviezdičiek nakreslí kríž

pre **dlzka**: 3

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

Príklad: Hviezdičkovanie 3

```
#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 šachovnicu

pre **r**: 10

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

Príklad: Hviezdičkovanie 4

pomocou hviezdičiek nakreslí "šachovnicu"

```
...
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

pre `r`: 14

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

Príklad: Hviezdičkovanie 5

```
#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

vnútorné zátvorky nemusia byť

pre **r**: 4

```
....
.**.
.**.
....
```

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;
}
```

Čo vypíše program?

```
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
```

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 musí 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;
}
```

Príkaz `switch`

- ak je viac hodnôt, pre ktoré chceme vykonať rovnaký príkaz (napr. hodnoty **`h_1`**, **`h_2`**, **`h_3`**):

```
switch (vyraz) {
    case h_1 :
    case h_2 :
    case h_3 : prikaz_123; break;
    case h_4 : prikaz_4; break;
     default : prikaz_def; break;
}
```

⇒ ak nie je vetva ukončená príkazom **`break`**, program neopustí **`switch`**, ale spracováva nasledujúcu vetvu v poradí - až po najbližšie **`break`**, alebo konca **`switch`**

Príkaz `switch`: príklad

časť programu vypíše znaky **123** po stlačení klávesy **'a'**, **'b'** alebo **'c'**. Po stlačení **'d'** vypíše **23** a po stlačení inej klávesy vypíše len **3**

```
switch (getchar()) {
   case 'a' :
   case 'b' :
   case 'c' : putchar(1);
   case 'd' : putchar(2);
   default  : putchar(3);
}
```

Príkaz `switch`: príklad

časť programu vypíše znaky `1` po stlačení klávesy `'a'`, `'b'` alebo `'c'`. Po stlačení `'d'` vypíše `2` a po stlačení inej klávesy vypíše len `3`

```
switch (getchar()) {

   case 'a' :
   case 'b' :
   case 'c' : putchar(1); break;
   case 'd' : putchar(2); break;
   default  : putchar(3); break;
}
```

Príkaz `switch`: poznámky

- príkaz **`break`**
  - ruší najvnútornejšiu slučku cyklu, alebo
  - ukončuje príkaz **`switch`**

  treba dávať pozor na cyklus vo vnútri **`switch`** a naopak

- vetva **`default`** nemusí byť ako posledná, z konvencie sa tam dáva
- ak je vetva **`default`** na konci, nie je **`break`** nutný, dáva sa z konvencie

Príkaz `switch`: príklad

Aj keď nie je **`default`** na konci, vykoná sa vtedy, keď nie je splnená žiadna iná vetva

```
switch (getchar()) {

   default :
      printf("Nestlacil si ani '1' ani '2'.\n");
      break;

   case '1' :
      printf("Stlacil si '1'.\n");
      break;

   case '2' :
      printf("Stlacil si '2'.\n");
      break;
}
```

Príkaz `switch`: príklad

```
int c = 0;
while (c != '*') {
    switch (c = getchar()) {
        case ' '  :
        case '\t' :
            putchar('#');
            continue;

        case '*'  :
            break;

        default   :
            putchar(c);
            break;
    }
}
```

Mieša veci, ktoré nemajú nič spoločné!!!

Príkaz `switch`: príklad

```
while ((c = getchar())
          != '*') {
   switch (c) {
      case ' '  :
      case '\t' :
         putchar('#');
         break;

      default   :
         putchar(c);
         break;
   }
}
```

časť programu číta znaky a opisuje ich na obrazovku, biele znaky nahradí `'#'` a po prečítaní `'*'` skončí -

lepšie riešenie

Príkaz `switch`: príklad

```
while ((c = getchar())
          != '*') {
    switch (c) {
       case ' '  :
       case '\t' :
                 c = '#';
       default   :
                 putchar(c);
    }
}
```

časť programu číta znaky a opisuje ich na obrazovku, biele znaky nahradí '#' a po prečítaní '*' skončí -

kratšie, ale menej prehľadné

Príkaz `goto`

- príkazu **`goto`** sa dá v štrukturovanom jazyku (ako je jazyk C) vždy vyhnúť → nepoužívať!

⇒ ak ho niekto chce používať - musíte si ho naštudovať

(Herout, 3. vydanie str. 59; Herout, 4. vydanie, str. 65)

Príkaz `return`

- ukončí práve sa vykonávajúcu funkciu, ktorá ho obsahuje
- vo funkcii **`main()`** - ukončí sa program
- často sa pomocou **`return`** vracia hodnota → neskôr

```
void vypis(int k) {

   if (k == 0)
      return;

   ... /* vypocet, kde je treba
          nenulove cislo */
   printf("k: %d, ...", k);
}
```

Časté chyby

```
if (i == 1) then
    ...
```

then nie je kľúčové slovo jazyka C

```
if i == 1
```

chýbajú zátvorky

```
if (i == 1)
    y = x
else
    x++;
```

chýba bodkočiarka

```
if (i = 1)
```

priradenie (=) namiesto porovnania (==)

Časté chyby

```
if (c = getchar() == '*')
```

chýbajú zátvorky

```
while (x == 1) do
```

za **while** nie je **do**

```
for (i = 0; i < 10; i++);
    x += i;
```

nemá tu byť bodkočiarka

Uvedomte si

- operátor priradenia: **`=`**
- operátor pre porovnanie: **`==`**
- logické **`&&`** (AND) a **`||`** (OR) majú skrátené vyhodnocovanie
- pre ukončenie slučky cyklu - príkaz **`break`**
- za každou vetvou príkazu **`switch`** musí byť **`break`** - ak nie je, vetvy musia súvisieť
- ak si nie ste istí s prioritami, zátvorkujte
- vyhýbajte sa podozrivým a komplikovaným kódom

Príklad: `switch`

Čo vypíše program?

```
#include <stdio.h>

int main() {
   int i;

   for(i=3; i<13; i++)
      switch(i) {
         case 3: printf("Hodnota je 3\n");
            break;
         case 4: printf("Hodnot
            break;
         case 5:
         case 6:
         case 7:
         case 8: printf("Hodnot
            break;
         case 11: printf("Hodnc
            break;
         default: printf("Nedef
      }
   return 0;
}
```

```
Hodnota je 3
Hodnota je 4
Hodnota je medzi 5 a 8
Hodnota je medzi 5 a 8
Hodnota je medzi 5 a 8
Hodnota je medzi 5 a 8
Nedefinovana hodnota
Nedefinovana hodnota
Hodnota je 11
Nedefinovana hodnota
```

program náhodne vyberie magické číslo. Používateľ číslo háda pokým ho nezistí alebo neukončí program načítaním čísla -1.

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
   int magicke;
   int tip;
   magicke = rand();

   printf("Vas tip na magicke cislo: ");
   scanf("%d", &tip);

   while (tip != magicke && tip != -1) {
      if (tip > magicke)
         printf("\ntip je prilis vysoky, zadajte dalsi: ");
      else printf("\ntip je prilis nizky, zadajte dalsi:\n");
      scanf("%d", &tip);
   }
   if (tip == magicke ) {
      printf("*** BINGO! ***");
      printf(" %d je magicke cislo.\n", magicke);
   }
   else
      printf("Skus nabuduce.\n");
   return 0;
}
```

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 vykreli 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: trojuholník – pridaný `for` (4)

Nakreslite trojuholníky vedľa seba (4x) aj pod seba (2x)

```
Pre n=5:
*     *     *     *
**    **    **    **
***   ***   ***   ***
****  ****  ****  ****
**********************
*     *     *     *
**    **    **    **
***   ***   ***   ***
****  ****  ****  ****
**********************
```

Ďalšie príklady

- Od zadaného písmena vypísať písmená do konca abecedy
- Od zadaného písmena vypísať k písmen
- Načíta sa celé číslo n zadá sa n čísel a vypíše sa ich súčet
- Vypísať čísla od 1 po zvolené n vždy oddelené znakom + (a medzerami) a na konci riadku pridať = a ich súčet
- n riadkov: súčet od 1 po 1, v ďašom riadku od 1 po 2, ..., až od 1 po n
- Vypísať čísla od 1 po zvolené n tak, že sa vypisujú len čísla deliteľné daným celým číslom k

Ďalšie príklady

- Nakresliť pomocou hviezdičiek obrázok X
- Obázok X opakovať n-krát pod seba
- Obrázok X opakovať m-krát vedľa seba