# Větvení a cykly

Úvod do programování 1

Tomáš Kühr

# Konstrukce if

- Příkazy se provádějí pouze při splnění dané podmínky
- Podmínka = jakýkoli logický výraz = cokoli celočíselného
- Volitelně i příkazy prováděné při nesplnění podmínky (za `else`)
- Typický zápis:

```
if (podmínka)
{
    příkazy 1
    ...
}
else
{
    příkazy 2
    ...
}
```

# Příklady if

```
if (hodnota < 0)
{
    hodnota = -hodnota;
    zmeneno = 1;
}
else
{
    zmeneno = 0;
}
```

```
if (hodnota < 0)
{
    hodnota = -hodnota;
    zmeneno = 1;
}
else
    zmeneno = 0;
```

```
if (hodnota < 0)
    hodnota = -hodnota;
```

# If v jiném if

- Relativně běžné
- Příklad:

```
if (hodnota < 0)
{
    printf("Záporné");
}
else
{
    if (hodnota > 0)
    {
        printf("Kladné");
    }
    else
    {
        printf("Nula");
    }
}
```

```
if (hodnota < 0)
    printf("Záporné");
else
    if (hodnota > 0)
        printf("Kladné");
    else
        printf("Nula");
```

# Konstrukce switch

- Větvení podle hodnoty celočíselného výrazu
- Příkazy pro určitou hodnotu výrazu (tj. pouze podmínky ve tvaru rovnosti)
- Default větev se provádí, pokud nebyla žádná podmínka splněna
- Typický zápis:

```
switch (výraz) {
    case hodnota1:
        příkazy 1
        ...
        break;
    case hodnota2:
        příkazy 2
        ...
        break;
    ...
    default:
        příkazy
        ...
        break;
}
```

# Příklady switch

```
int cislo;
char *text;
...
switch (cislo)
{
    case 1:
        text = "Jedna";
        break;
    case 2:
        text = "Dva";
        break;
    default:
        text = "Nevim";
        break;
}
printf(text);
```

```
char znak, posunout, novy_znak;
int posun = 2;
...
switch (posunout){
    case 'Y':
        novy_znak = znak + posun;
        posunout = 'N';
        break;
    case 'N':
        novy_znak = znak;
        break;
    default:
        printf("Chyba!\n");
}
```

# Switch – větev pro více hodnot

- Není úplně typické, ale může se hodit
- Několik návěstí case za sebou
- Větev společná pro více uvedených hodnot
- Příklad:

```
switch (cislo)
{
    case 1:
    case 2:
        text = "Jedna nebo Dva";
        break;
    default:
        text = "Nevim";
        break;
}
```

# Příklady složitějšího větvení

- Switch v if:

```
if (slovne){
    switch (cislo){
        case 1:  printf("Jedna");
                 break;
        case 2:  printf("Dva");
                 break;
        case 3:  printf("Tri");
                 break;
        default: printf("Neumim");
                 stop = 1;
                 break;
    }
} else {
    printf("%d\n", cislo);
}
```

# Příklady složitějšího větvení

- If ve switch:

```
switch (akce) {
    case 'n':
        vysledek = vstup1 * vstup2;
        break;
    case 'd':
        if (vstup2 == 0) {
            printf("Nelze delit nulou.\n");
            break;
        } else {
            vysledek = vstup1 / vstup2;
            break;
        }
    default:
        printf("Neznama akce.\n");
        break;
}
```

# Motivace k cyklům

- Vypište prvních 5 přirozených čísel

```
printf("1\n");
printf("2\n");
printf("3\n");
printf("4\n");
printf("5\n");
```

- Vypište prvních n přirozených čísel

```
int n;
scanf("%d", &n);

if (n >= 1) printf("1\n");
if (n >= 2) printf("2\n");
if (n >= 3) printf("3\n");
if (n >= 4) printf("4\n");
if (n >= 5) printf("5\n");
...
```

# Cyklus while

- Opakování bloku příkazů
- Dokud je splněna uvedená podmínka
- Podmínku píšeme na začátek cyklu
- Testuje se na začátku každého průchodu cyklem
- Typický zápis:

```
while (podmínka)
{
    příkazy
    ...
}
```

# Příklady while cyklu

```c
int n, i;

scanf("%d", &n);

i = 1;
while (i <= n){
    printf("%d\n", i);
    i++;
}
```

```c
int prvni, druhe;
int vetsi, mensi;
...
vetsi = prvni;
mensi = druhe;
while (mensi != 0){
    int zbytek;
    zbytek = vetsi % mensi;
    vetsi = mensi;
    mensi = zbytek;
}

printf("Vysledek je %d. \n", vetsi);
```

# Cyklus for

- Složitější konstrukce cyklu
- Podporuje nejběžnější způsob použití cyklu
  - Inicializace řídících proměnných před cyklem
  - Test podmínky na začátku cyklu
  - Změny řídících proměnných mezi průchody
- Typický zápis:

```
for (inicializace; podmínka; iterace)
{
    příkazy
    ...
}
```

# Příklady for cyklu

```c
int od, po;

scanf("%d", &od);
scanf("%d", &po);

for (int i = od; i <= po; i++)
{
    printf("%d\n", i);
}
```

```c
int od, krok;

scanf("%d", &od);
scanf("%d", &krok);

for (int i = od; i > 0; i-= krok)
{
    printf("%d\n", i);
}
```

# For vs. while cyklus

- Cyklus for lze  přepsat pomocí while
- For cyklus:

```
for (inicializace; podmínka; iterace)
{
    příkazy
    ...
}
```

- Odpovídá:

```
inicializace;
while (podmínka)
{
    příkazy
    ...
    iterace;
}
```

# Největší společný dělitel (pomocí for)

```
int prvni, druhe;
int vetsi, mensi;

...

for (vetsi = prvni, mensi = druhe ; mensi != 0; )
{
    int zbytek;
    zbytek = vetsi % mensi;
    vetsi = mensi;
    mensi = zbytek;
}

printf("Vysledek je %d. \n", vetsi);
```

# Cyklus do-while

- Na rozdíl od předchozích typů cyklu
- testuje podmínku až na konci prvního průchodu.
- Příkazy v cyklu se tedy vždy minimálně 1 krát vykonají.
- Typický zápis:

```
do
{
    příkazy
    ...
} while (podmínka);
```

# Příklady cyklu do-while

```c
int n, i;

scanf("%d", &n);

i = 1;
do
{
    printf("%d\n", i);
    i++;
} while (i <= n);
```

```c
int prvni, druhe;
int vetsi, mensi;
...
vetsi = prvni;
mensi = druhe;
do
{
    int zbytek;
    zbytek = vetsi % mensi;
    vetsi = mensi;
    mensi = zbytek;
} while (mensi != 0);

printf("Vysledek je %d. \n", vetsi);
```

# Přerušení cyklu

- Cykly bývají v reálných programech i složitější
- Někdy potřebujeme přerušit cyklus i jinde než na začátku/konci
- Často v závislosti na různých podmínkách
- Příkazy pro přerušení cyklu:
  - okamžité vyskočení - `break;`
  - přerušení aktuálního průchodu a přechod k dalšímu - `continue;`
- Vztahují se vždy k "nejbližšímu" cyklu

# Příklad přerušení break

```
int cislo;
int mocnina;
int min = 100;

for (cislo = 1; ; cislo++)
{
    mocnina = cislo * cislo;
    if (mocnina > min)
    {
        printf("%d\n", mocnina);
        break;
    }
}
```

# Příklad přerušení continue

```
int pocet = 5;
int mocnina = 1;
int cislo = 1;
int max = 500;

while (mocnina < max)
{
    printf("%d, ", mocnina);
    cislo++;
    mocnina = cislo * cislo;
    if ((cislo-1) % pocet != 0) continue;
    printf("\n");
}
```

# Složitější zacyklení

- Jeden cyklus může (samozřejmě) být uvnitř druhého cyklu:

```
int pocet = 10;
int delka_radku = 3;

for (int i = 0; i < pocet; i++)
{
    for (int j = 0; j < pocet; j++)
    {
        printf("%d * %d = %d, \t", i, j, i * j);
        if ((i * pocet + j + 1) % delka_radku != 0)
            continue;
        printf("\n");
    }
}
```

- Příkazy pro přerušení cyklu se týkají vždy toho nejvnitřnějšího cyklu, ve kterém jsou uvedeny.

# Větvení programu – cvičení 4

Vytvořte program, který po načtení data (den. měsíc. rok) zkontroluje a oznámí, zda daný den skutečně existuje (datum je správně) nebo ne. Roky dovolte zadávat od 1582 po 2017, měsíce 1 až 12 a dny podle skutečného počtu dní v měsíci. V jednodušší variantě můžete počítat s 28 dny v únoru, ve složitější si pak vyhrajte i s přestupnými roky...

Příklady použití:

```
Zadej datum: 23. 4. 1983
Den existuje.

Zadej datum: 29. 2. 2000
Den existuje.

Zadej datum: 29. 2. 1900
Den neexistuje.
```

# Cykly – cvičení 4

Vytvořte program, který od uživatele načte první člen, diferenci (resp. kvocient) a počet členů aritmetické (resp. geometrické) posloupnosti a poté tuto posloupnost vypíše na obrazovku.

Příklad použití pro aritmetickou posloupnost:

```
Zadejte prvni clen: 3
Zadejte diferenci: 2
Zadejte pocet: 5
Posloupnost: 3, 5, 7, 9, 11
```