

## 3 – Řídící struktury

### Příklad 3.1

Jaká bude výstupní hodnota proměnné suma?

```
#include<stdio.h>
main()
{
    int i, suma = 0;
    for(i=0;i<10;i++) {
        switch(i) {
            case 0:
            case 2:
            case 3:
            case 5:
                suma++;
            default:continue;
            case 4:
                break;
        }
        break;
    }
    printf ("%d",suma);
}
```

### Příklad 3.2

Následující program je ukázkou toho, kterak program nemá rozhodně vypadat. Proto ho přepište bez break, continue a goto.

```
/* Spočte pocet zadanych 'a' */
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>

int main( void )
{
    int pocet_a = 0;
    char c;
    c = getchar();
    while (1)
    {
        if (c == '\n')
            break;
        if (isdigit( c ))
            continue;
        if (c == 'a')
            goto pridej_a;
        get_next_char:
        c = getchar();
        goto konec;
        pridej_a:
        pocet_a++;
        goto get_next_char;
        konec: ;
    }
    printf("%d",pocet_a);
    exit( 0 );
}
```

### Příklad 3. 3 a)

Následující program ukazuje praktické použití všech probraných příkazů (program byl s úpravami převzat z lit.2). Program pracuje jako jednoduchá kalkulačka. Je-li zadáno první číslo, libovolný počet mezer, operátor, libovolný počet mezer a druhé číslo, spočte výsledek příslušné matematické operace. Při špatně zadaném operátoru program hlásí chybu. Použité funkce jsou umístěny v souboru uzitek.h. Vyzkoušejte funkci programu také při špatně zadaném čísle (např. 1E3 \* 123).

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

int main( void )
{
    extern double parse_num(), evaluate(); /* použite funkce */
    extern void skip_spaces(); /* použite funkce */
    double op1, op2, answer;
    int operator;

    printf( "Enter <number> <op> <number><newline>: ");
    skip_spaces(); /* preskacu vsechny mezery */
    op1 = parse_num(); /* nactu cislo */
    skip_spaces(); /* preskacu vsechny mezery */
    operator = getchar(); /* nactu operator */
    skip_spaces(); /* preskacu vsechny mezery */
    op2 = parse_num(); /* nactu cislo */
    answer = evaluate( op1, op2, operator );
    printf( "%f\n", answer ); /* tisknu vysledek */
    exit( 0 );
}
```

### Příklad 3.3 b)

toto je obsah souboru uzitek.h

```
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
#define DECIMAL_POINT '.' /* mohu menit tecku za carku */

double parse_num()
{
    int c, j, digit_count = 0;
    double value = 0, fractional_digit;

    while (isdigit( c = getchar() ) ) /* zpracuju cast vlevo od tecky */
    {
        value = value * 10;
        value = value + (c - '0');
    }

    /* Kdyz neni c cislice, podivam se, zda to neni desetinná tečka */
    if (c == DECIMAL_POINT) /* get fraction */
        while (isdigit( c = getchar() ) ) /* zpracuju desetinnou cast */
        {
            digit_count++;
            fractional_digit = c - '0';
            for (j=0; j < digit_count; j++)
                fractional_digit = fractional_digit/10;
            value = value + fractional_digit;
        }
}
```

```
    }
    ungetc( c, stdin ); /* znak, který už není mezerou, vrátím do bufferu */
    return value;
}
```

```
void skip_spaces(void)
{
    int c;
    for (c = getchar(); isspace( c ); c = getchar())
        /* prázdný příkaz */
        ungetc( c, stdin ); /* znak, který už není mezerou, vrátím do bufferu */
}
```

```
/* Tisknu zprávu, o jakou jde chybu. Funkce je typu void neboť nevrací hodnotu, pouze něco dělá
*/
```

```
#define ERR_INPUT_VAL 1
#define ERR_OPERAND 2
#define ERR_OPERATOR 3
#define ERR_TYPE 4
```

```
void print_error( int error_code )
{
    switch (error_code)
    {
        case ERR_INPUT_VAL:
            printf("Error: Illegal input value.\n");
            break;
        case ERR_OPERAND:
            printf("Error: Illegal operand.\n");
            break;
        case ERR_OPERATOR:
            printf("Error: Unknown operator.\n");
            break;
        case ERR_TYPE:
            printf("Error: Incompatible data.\n");
            break;
        default: printf("Error: Unknown error code %d\n",
                        error_code);
            break;
    }
}
```

```
/* Tato funkce vyhodnotí výraz, jsou-li zadány dva operandy a operátor */
```

```
double evaluate( double op1, double op2, char operator )
{
    extern void print_error ();

    switch (operator)
    {
        case '+': return op1 + op2;
        case '-': return op1 - op2;
        case '*': return op1 * op2;
        case '/': return op1 / op2;
        default : /* Illegal operator */
            print_error( ERR_OPERATOR );
            exit( 1 );
    }
}
```

Popis obsažených funkcí v uzitek.h a úkoly pro řešení:

- funkce `parse_num` umí přečíst řadu číslic včetně desetinné tečky a přeměnit ji v číslo. Dobře si povšimněte zejména do sebe vložených příkazů.
- funkce `skip-spaces` přeskáče všechny načtené znaky, které jsou mezera. Je užít cyklus `for`, který nedělá nic. Poslední načtený znak, který už je nemezera, je vrácen do bufferu - to je častá finta, neboť by se jinak znak poztrácel.  
Tento program přepište pomocí cyklu `while`.
- `print_error` jednoduchá funkce, která napíše hlášku podle kódu, který je jí poslán.
- funkce `evaluate` : vyhodnotí matematický výraz dle zadaného operátoru. Doplňte tuto funkci tak, aby při zadaném znaku `M` vypsala větší z obou čísel a při `N` menší číslo. Jako další rozšíření doplňte pro `P` výpočet  $x^y$ . Pro tuto operaci je v C funkce `pow()`. Přesný popis si najděte v helpu. Je nutno užít `math.h`.
- Vzhledem k tomu, že `scanf()` již umí jak přeskákat mezery, tak zpracovat desetinné číslo, zjednodušte program za využití této funkce.
- Dále zcela vynechte použití proměnné `answer`.
- Program obsahuje řadu operací, které jsou částí tzv. standardních knihoven jazyka C. Prohlédněte si jejich popis v helpu.

### Příklad 3.4

Následující program ukazuje použití nekonečné smyčky. Jedná se opět o předcházející kalkulátor, nyní s užitím funkce `scanf()`, tedy v mnohem jednodušší podobě. Program počítá stále další a další výpočty. Cyklus by měl být zakončen např. `CTRL-C` nebo `CTRL- break`. Nutno ovšem dopředu vyzkoušet, zda tyto pokyny program skončí. Např. autorka tohoto textu musela provést reset.

```
#include <ctype.h>
#include <stdio.h>
#include "uzitek.h"

int main( void )
{
    double op1, op2, answer;
    extern double evaluate();
    char operator;

    while (1)
    {
        printf( "Enter <number> <op> <number>"
            "<newline>: ");
        scanf( "%lf %c %lf", &op1, &operator, &op2 );
        answer = evaluate( op1, op2, operator );
        printf( "%f\n", answer );
    }
}
```

### Příklad 3.5

Napište program, který rozčlení přečtené znaky do následujících skupin:

```
ZNAMENKA ( " , ? ! : ; ( ) . ' )
CIFRY ( '0' - '9' )
PISMENA ( 'a' - 'z', 'A' - 'Z' )
BILE_ZNAKY ( mezera, '\n', '\r', '\t' )
OSTATNI ( cokoliv jineho )
```

Napište dvě verze : první se `switch` a druhou s `if, else` a `return`. Porovnejte délku a přehlednost programu.

### Příklad 3.6

Ověřte na následujícím programu podivné chování při špatně zadané podmínce pro větvení (if). Zatímco větvení by mělo fungovat takto: pro čísla mezi 3 a 10 - výpis "dobrá hodnota" a pro ostatní výpis "špatná hodnota", program se chová zcela jinak. Přepište podmínku pro if tak, aby program dělal , to co má.

```
#include<stdio.h>
main()
{
int cislo;
printf("zadej cislo mezi 3 a 10");
scanf("%d",&cislo);
if (3 < cislo< 10) printf ("dobra hodnota");
else printf(" spatna hodnota");
}
```

### Příklad 3.7

Napište program, který vytiskne znaky od 'a' do 'z' a od 'A' do 'Z' a jejich číselnou hodnotu ( desítkově).

### Příklad 3.8

Napište předcházející program, který vytiskne číselnou hodnotu desítkově, šestnáctkově a osmičkově.

### Příklad 3.9

Co dělá tento program ?

```
#include <stdio.h>
main()
{
int i;
printf("\n Test prikazu break a continue \n");
for (i = 1; i < 20; i++) {
printf("\n%d ", i);
if (i >= 15) {
printf("Break\n");
break;
}
if (i >= 10) {
printf("Continue ");
continue;
}
printf("Posledni prikaz cyklu for");
}
}
```

### Příklad 3.10

Napište program, který přečte z klávesnice číslo min, poté číslo max a poté číslo n ( všechna celá) . Program přečte z klávesnice n celých čísel ( užit je cyklus for) a spočítá, kolik čísel je z intervalu <min,max>. Program upravte tak, aby uživatel programu stále věděl, co má dělat.

### Příklad 3.11

Napište program, který načítá celá čísla až '\n' a spočte , kolik z nich je dělitelné osmi ( sedmi, třema atd.).

### Příklad 3.12

Napište program za pomoci cyklu for, který vypíše na obrazovku tabulku mocnin dvou (tří, čtyř atd. ) od  $2^1$  do  $2^{20}$  ve tvaru:

$2^1$	2
$2^2$	4
$2^3$	8

### Příklad 3.13

Přepište následující příkaz bez užití for a operátoru zvýšení hodnoty:

```
for (n = A; n < B ; n++) sum += n;
```

### Příklad 3.14

Co udělá následující cyklus aplikovaný na řetězec?

```
while (*char_pointer ++);
```

### Příklad 3.15

Následující cyklus kopíruje znakový řetězec na jiné místo. Vysvětlete, jak to provádí.

```
while ( * dest_pointer++ = * source_pointer++);
```

### Příklad 3.16

Co dělá následující funkce? Nakreslete její průběh.

```
int signum(x)
{
    int x;
    {
        if (x > 0) return 1;
        else if (x < 0) return -1;
        else return 0;
    }
}
```

### Příklad 3.17

Každý počítač má určitou omezenou přesnost pro reprezentaci čísel typu float a double . Pro určité velmi malé epsilon začne platit:

$$1.0 = 1.0 + \text{epsilon}$$

Napište 2 programy, které najdou největší takové epsilon pro váš počítač. Jeden program pro floats a jeden pro doubles. V předcházejícím výrazu užíjte 1.0, jak uvedeno, a ne 0.0, neboť tento výraz mnoho počítačů vyhodnocuje jiným způsobem.

### Příklad 3.18

Napište funkci, která jako parametr má číslo a napíše tomu odpovídající počet mezer. S použitím této funkce napište program, který ze standardního vstupu čte znaky a opisuje je na standardní výstup. Tabulátor ale nahrazuje 5 mezerami.

### Příklad 3.19

Mnoho programů, které musí přemísťovat data, používá tzv. kontrolní součet. Užití tohoto kontrolního součtu zvyšuje pravděpodobnost, že data byla přesunuta korektně. Obvyklou technikou je sečtení selkové hodnoty ( sumy) všech znaků. Tj. např. byla poslány někam znaky 'e', 'f', 'g'. Celkový součet je poté 101 +102 +103 tj. 306. Zásilající stranou je poté spolu se znaky zaslán kontrolní součet. Přijímající strana z přijatých znaků také spočte kontrolní součet a porovná obě hodnoty. Napište funkci kontr\_suma(), která umí spočítat kontrolní součet ze znaků zadaných z klávesnice.

**Od RNDr. Dagmar Brechlerové.**