Programovací jazyk C: Procedury a funkce

Programování F1400 + F1400a

doc. RNDr. Petr Mikulík, Ph.D.

podzimní semestr 2016; bonus od 2017; aktualizace 2019

```
void uvod ()
{ printf("Vitej!\n"); }

void chyba_konec ( int n )
{ printf("Chyba %i\n", n); exit(1); }

int den_v_tydnu_nedele ()
{ return 7; }

double muj_polynom ( double x )
{
    return 3*x*x + 4.2*x*x - 3*x - 1.5;
}
```

```
double faktorial1 (int n)
{
    int k;
    double f = 1;
    if (n <= 1) return 1;
    for (k=2; k<=n; k++) f *= k;
    return f;
}
double faktorial2 (int n)
{
    if (n <= 1) return 1;
    return n * faktorial2(n-1);
}
```

Často používané kusy kódu

- Dosud byl veškerý náš kód napsaný v sekci `int main () { ... }`, ale často se hodí rozdělit program na části.
- Často opakované činnosti, tedy kusy kódy – v rámci jednoho programu nebo i ve více programech), např. sqrt(x), exp(x), výpis chyb, ...– přesuneme jinam, na vyžádání se vykoná daná činnost a vrátí se výsledek $\Rightarrow$
  - zpřehlednění a uspořádání kódu,
  - při ladění či změnách stačí tento kód opravit na jediném místě.
- **Struktura programu:** zdrojový kód obsahuje **hlavičku** (`#include ...`) a poté posloupnost deklarací nebo definicí **globálních proměnných**, **funkcí** a **procedur**.
- **Funkce:** blok kódu, který vrací nějaký výsledek (např. derivaci nějaké funkce, počet malých písmen v řetězci, počet pulsů načtených z detektoru, inverzní matici, data načtená ze souboru, apod.).
- **Procedura:** blok kódu, který nic nevrací (může třeba vypsat nějakou zprávu na obrazovku, poslat soubor na server apod.).

Procedury a funkce

- Příklad procedur a funkcí s maximálně 1 parametrem:

```
1   void uvod ()
2   {
3       printf("Vitej v jednoduchem programu!\n");
4   }

5   void chyba_konec ( int n )
6   {
7       printf("S hodnotou n=%i program nemuze pocitat!\n", n);
8       exit(1);
9   }

10  int den_v_tydnu_nedele ()
11  {
12      return 7; // na kontinentu je nedele 7. den v tydnu
13      // return 1; // v Britanii je nedele 1. den v tydnu
14  }

15  double muj_polynom ( double x )
16  {
17      // return x*x*4 + 7.2*x*x - 6*x - 2.5; // nejaky muj polynom
18      return 4*x*x - 2*x + 1; // nejaky muj polynom
19  }
```

Procedury a funkce

- Kompletní příklad – program se dvěma procedurami:

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <math.h>

4   void uvod ()
5   {
6       printf("Vitej v jednoduchem programu!\n");
7   }



8   void chyba_konec ( double a )
9   {
10      printf("Pro hodnotu %g program nemuze logaritmus spocitat!\n", a);
11      exit(1);
12  }



13  int main ()
14  {
15      double x;
16      x = -1;   // zkuste zadat: -1, 0, nebo 1

17      uvod();

18      if (x == 0)
19          chyba_konec(x);
20      if (x < 0)
21          chyba_konec(x);

22      printf("logaritmus %g je %g\n", x, log(x));
23      return 0;
24  }
```

Lokální proměnné

- Platnost (životnost) proměnné je vždy v rámci jednoho bloku, tedy mezi `{ ... }`.

- Následující kód používající **lokální proměnnou** `y` vypíše čísla y od 1 do 11, a nakonec 123:

```
1     int x;
2     int y = 123;

3     for (x=0; x<=10; x++) {
4         int y = x + 1;
5         printf("x=%i  y=%i\n", x, y);
6     }

7     printf("Na konci je y=%g\n", y);
```

Globální vs lokální proměnné

- **Globální proměnné** jsou k dispozi všude od místa jejich deklarace.
- Příklad – součet řady s globálním omezením na velikost sčítaných prvků:

```
1   int max_prvek = 100;  // globalni promenna, je viditelna vsude

2   int soucet_rady (int prvek1, int prvek2)
3   {
4       int suma = 0;
5       if (prvek2 > max_prvek) {
6           printf("Takto velke prvky uz nemam pocitat.\n");
7           exit(1);
8       }
9      while (prvek1 <= prvek2) {
10         suma += prvek1;
11         prvek1++;
12     }
13     return suma;
14  }
```

- Pozn.: cyklus `while` v příkladu jde „zjednodušit“ až na

```
1       while (prvek1 <= prvek2) suma += prvek1++;
```

Funkce sinc() na tři různé způsoby

- Funkce $\operatorname{sinc} x = \frac{\sin x}{x}$ se používá např. v optice (difrakce světla na štěrbině). Funkce je spojitá, limitou spočteme hodnotu v nule $\operatorname{sinc} 0 = 1$.

```
1     double sinc1 (double x)  // program vyuzivajici pomocnou promennou
2     {
3         double y;
4         if (x==0) y = 1;
5         else y = sin(x)/x;
6         return y;
7     }

8     double sinc2 (double x)  // program s jednim if bez else
9     {
10        if (x==0) return 1;
11        return sin(x)/x;
12    }
```

- A lze to i na jeden řádek s použitím tohoto ternárního operátoru:

```
1     double sinc3 (double x)
2     {
3         return (x == 0) ? 1 : sin(x)/x;
4     }
```

Ternární operátor (...) ? ... : ...

- **Unární operátor** – jeden parametr: opačná hodnota $-x$, $\sin x$, $\cos x$, ...
- **Binární operátor** – dva parametry: sčítání, odčítání, násobení, x^y, ...
- **Ternární operátor** – tři parametry. Např.

  `(Podmínka) ? Výraz1 : Výraz2`

  jehož výsledkem je Výraz1 pokud Podmínka platí, jinak Výraz2.

  - Příklad – $\operatorname{sinc} x = \frac{\sin x}{x}$:

```
1    double sinc3 (double x)
2    {
3        return (x == 0) ? 1 : sin(x)/x;
4    }
```

  - Příklad – absolutní hodnota (jako je fabs(x)):

```
1        abs_x = ((x >= 0) ? x : -x);
```

  - Příklad – Heavisideova (skoková) funkce:

```
1    double Heaviside (double x)
2    {
3        return (x == 0) ? 0.5 : ((x > 0) ? 1 : 0);
4    }
```

Rekurentní funkce

- **Rekurentní funkce:** Funkce volající sama sebe.

  Pozor na zacyklení! Vždy nastavit okrajovou podmínku.
- Příklad – **výpočet faktoriálu**:
  - Přímý výpočet z definice $n! = \prod_{k=1}^{n} k$

```
1    double faktorial1 (int n)
2    {
3        int k;
4        double f = 1;
5        if (n <= 1) return 1;  // vysledek pro 0! nebo 1!
6        for (k=2; k<=n; k++)
7            f *= k;
8        return f;
9    }
```

  - Rekurentní výpočet z definice $n! = n \cdot (n-1)!,\ 1! = 0! = 1$

```
1    double faktorial2 (int n)
2    {
3        if (n <= 1) return 1;
4        return n * faktorial2(n-1);
5    }
```

- Ke každému rekurentnímu algoritmu existuje i přímý.
- Rekurentní algoritmy spotřebují hodně paměti.

int main() je také funkce

- Funkce main() je taktéž funkce; program v ní začíná a očekává se vrácení celého čísla:

```
1    int main ()
2    {
3        ...
4        return 0; // celociselna navratova hodnota
5    }
```

- Později si ukážeme, jak je možné předat programu parametry z příkazové řádky:

```
1    int main (int argc, char *argv[])
2    {
3        ...
4    }
```

Příkaz switch ... case

- Příkaz **switch** slouží k vykonání kódu z jedné větve podle hodnoty celočíselného výrazu. Je ekvivalentní vnoření více příkazů if – ale switch je mnohem přehlednější.

```
1   switch (CelociselnyVyraz) {
2       case Hodnota1:
3               Prikaz1a;
4               Prikaz1b;
5               ...
6               break;
7       case Hodnota2:
8               Prikaz2a;
9               Prikaz2b;
10              ...
11              break;
12      ...
13      default:
14              PrikazXa;
15              ...
16  }
```