KATEDRA INFORMATIKY
PŘÍRODOVĚDECKÁ FAKULTA
UNIVERZITA PALACKÉHO V OLOMOUCI

Rekurze

1. cvičení

Jiří Zacpal

KMI/ZP2 – Základy programování 2

Doporučená literatura

- Pavel Herout: Učebnice Jazyka C. Kopp, 2007.
- Reek Kenneth: Pointers on C. Addison Wesley, 1997.
- Robert Sedgewick: Algorithms in C. Addison-Wesley Professional, 2001.
- Jeri R. Hanly, Elliot B. Koffman: Problem Solving and Program Design in C. Addison Wesley, 2006.
- Eric S. Roberts: Programming Abstractions in C. Addison Wesley, 1997.
- Eric S. Roberts: The Art and Science of C. Addison Wesley, 1994.
- Libovolné další učebnice jazyka C

Požadavky na zápočet

- pro zápočet je potřeba získat **20 bodů**:
  - 1 bod na každém cvičení
  - 0 až 5 bodů za úkoly vyhlašované v semestru (celkem 2 úkoly)
  - 0, 3, 5 bodů za písemné práce v semestru (celkem 2 za semestr)
- individuální domluva možná

Konzultace

- v pracovně 5.044
- každý pátek 9.00 – 11.00
- jindy po vzájemné domluvě
- email: jiri.zacpal@upol.cz
- odkazy:
  - phoenix.inf.upol.cz/~zacpal/zp2.html
  - jazykc.inf.upol.cz
  - Vyuka\kmi_zp2

Funkce - opakování

- Co je to funkce?
- Co je parametr funkce?
- Jaký je rozdíl mezi předáním parametru hodnotou a odkazem (ukazatelem, adresou)?
- Jaké rozlišujeme rozsahy platnosti proměnných (resp. identifikátorů)? Jaké jsou mezi nimi rozdíly?

Rekurzivní funkce

- funkce, která ve svém těle volá sama sebe
- postupným voláním funkce je problém zjednodušován → dekompozice → metoda rozděl a panuj
- dvě části rekurze:
  - **limitní podmínka rekurze** je podmínka, po jejímž splnění je vyhodnocen výraz, jež nezpůsobí další aplikaci samotné rekurzivní funkce
  - **předpis rekurze** je část těla funkce, při jejímž vyhodnocení dochází k rekurzivní aplikaci funkce

Průběh výpočetního procesu rekurzivní funkce

- výpočetní proces se skládá ze dvou fází:
  - **fáze navíjení**
    - je fáze, ve které dochází k postupné rekurzivní aplikaci
  - **fáze odvíjení**
    - nastává po dosažení limitní podmínky rekurze
    - během této fáze dochází k dokončení vyhodnocenı těla procedury
  - příklady: funkce pro výpočet faktoriálu, funkce pro výpočet fibonacciho čísla, quicksort, ...

Výhody a nevýhody rekurze

```
long double factrec(unsigned int n){
    if (n==0) return 1;
    return n * factrec(n-1);

}

long double factiter(unsigned int n){
    long double out = 1;
    unsigned int i;
    for (i=2; i<=n; i++) out *= i;
    return out;

}
```

Úkol

Fibonacciho čísla

- Napište v jazyku C rekurzivní a nerekurzivní funkci pro výpočet *n*-tého fibonacciho čísla. Porovnejte rychlosti výpočtu těchto dvou funkcí pro větší *n*.
- Připomínáme, že posloupnost fibonacciho čísel je definována rekurzivně:

  Fib(0) = 0

  Fib(1) = 1

  Fib(n) = Fib(n-1) + Fib(n-2) pro n > 1

Příklad výstupu:

Fib(8) = 21

Fib(24) = 46368

Fib(35) = 9.22747E+006

Úkol – řešení (rekurze)

```
double fibonaccirec(int n)
 {
    if (n<2) return n;
        else
    return fibonaccirec(n-
1)+fibonaccirec(n-2);
 }
```

Úkol – řešení (iterační)

```
double fibonacciiter(int n)
 {
        double fib=0,Nmin1,Nmin2,temp;
        if(n<2) return n;
        Nmin1=1;
        Nmin2=0;
        for (int i=2;i<n;i++)
        {
                temp=Nmin1;
                Nmin1=Nmin1+Nmin2;
                Nmin2=temp;
         }
         return (Nmin1+Nmin2);
 }
```

Bodovaný úkol

Napište v jazyku C rekurzivní funkci
`int puleni(int cisla[], int a, int b, int hledane)`,
která pomocí metody půlení intervalu najde v zadaném setříděném poli `cisla` hodnotu `hledane` a vrátí její index v tomto poli. Připomínáme, že metoda půlení intervalu je založena na porovnání hodnoty hledaného čísla s číslem "uprostřed" právě prohledávaného intervalu (v našem případě intervalu mezi prvky s indexy `a` a `b`). Pokud se hodnoty rovnají, našli jsme hledané číslo a můžeme tedy přímo vrátit jeho index. Pokud se nerovnají, stačí (rekurzivním voláním) prohledávat pouze jeden z intervalů prvek s indexem `a` až "prostřední prvek" nebo "prostřední prvek" až prvek s indexem `b`.

- Zdrojový soubor: up2_ukol_01_puleni.cpp