Úvod

Karel Richta a kol.

katedra počítačů FEL ČVUT v Praze

Přednášky byly připraveny s pomocí materiálů, které vyrobili Ladislav Vágner, Pavel Strnad, Martin Hořeňovský, Aleš Hrabalík

Programování v C++, A7B36PJC
09/2015, Lekce 1

https://cw.fel.cvut.cz/wiki/courses/a7b36pjc/start

Programy a programovací jazyky

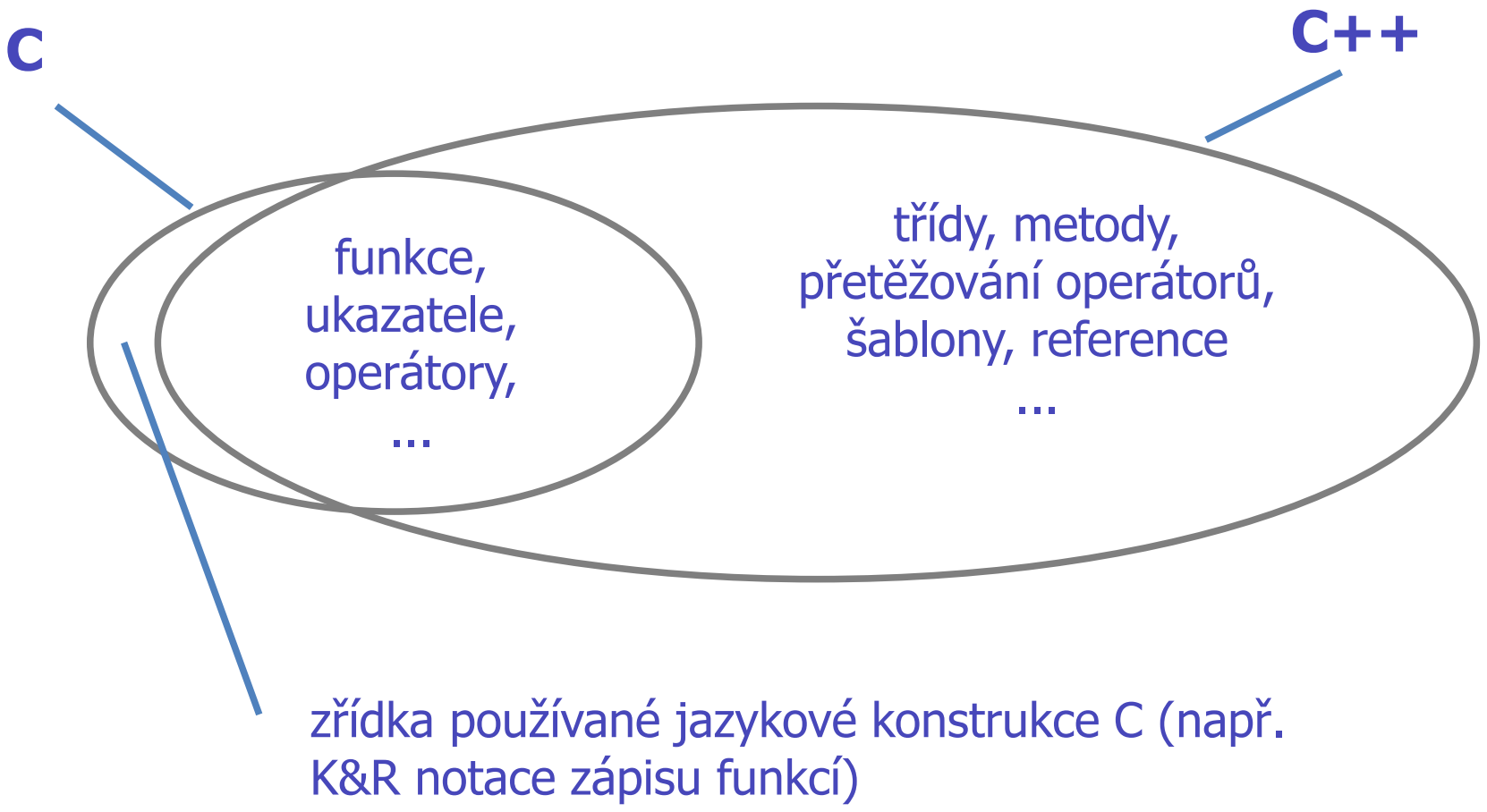
Program je předpis pro provedení určitých akcí počítačem zapsaný v programovacím jazyku (je to vždy zápis algoritmu?).

- Programovací jazyky:
  - strojově orientované (strojový jazyk = jazyk fyzického procesoru),
  - asembler (jazyk symbolických adres),
  - vyšší jazyky:
    - **imperativní** (příkazové, procedurální),
    - **neimperativní** (např. funkcionální, logické).
- Hlavní rysy imperativních jazyků (např. C, C++, Java, Pascal, Basic, atd.):
  - zpracovávané údaje mají formu **datových objektů** různých typů, které jsou v programu reprezentovány pomocí **proměnných** resp. **konstant**, uložených v **paměti** počítače,
  - **program** obsahuje deklarace a příkazy,
  - **deklarace** definují význam jmen (identifikátorů),
  - **příkazy** předepisují akce s datovými objekty nebo způsob řízení výpočtu.

Historie C a C++

- Jazyk C původně vznikl jako systémový jazyk pro operační systém Unix (aby se nemusel psát v asembleru).
- Vycházel z jazyka BCPL (Martin Richards, Cambridge).
- Původní verze: jazyk B (Ken Thompson, PDP-7, 1970).
- Při přechodu na PDP-11: jazyk C (Dennis Ritchie a spol. - 1972)
- 1978 - Kernighan, Ritchie: The C Programming Language (K&R C)
- 1982 - ANSI C - X3J11 (ISO WG14 1990, později doplňky ISO/IEC 9899, 1999 tzv. ISO C99).
- Později vznikají různá objektová rozšíření C (Objective C, C++, ..).
- C++ navrhl Bjarne Stroustrup, 1983, AT&T Bell Laboratories.
- 1985 Bjarne Stroustrup: The C++ Programming Language.
- 1990 Bjarne Stroustrup: The Annotated C++ Reference Manual.
- 1998 - ISO WG21, ISO/IEC 14882 (ANSI C++ - X3J16, 1998).
- 2011 - Nové doplňky standardů C11, C++11, C++14.

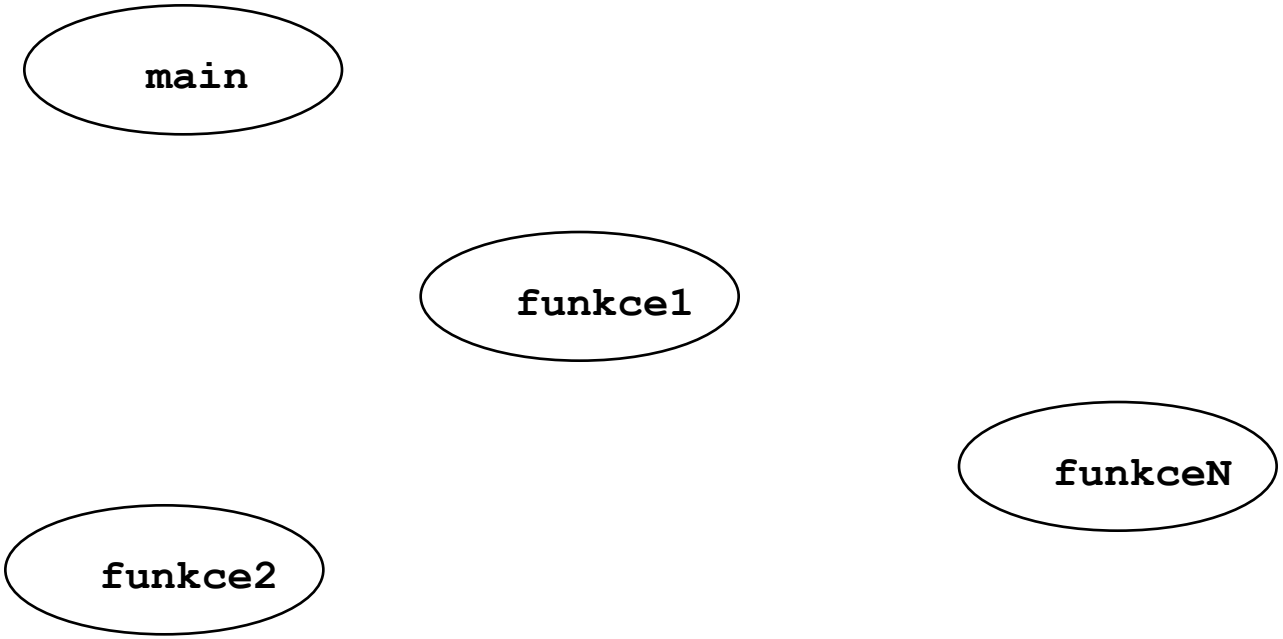
Vztah C a C++


C
C++
funkce,
ukazatele,
operátory,
...
třídy, metody,
přetěžování operátorů,
šablony, reference
...
zřídka používané jazykové konstrukce C (např.
K&R notace zápisu funkcí)

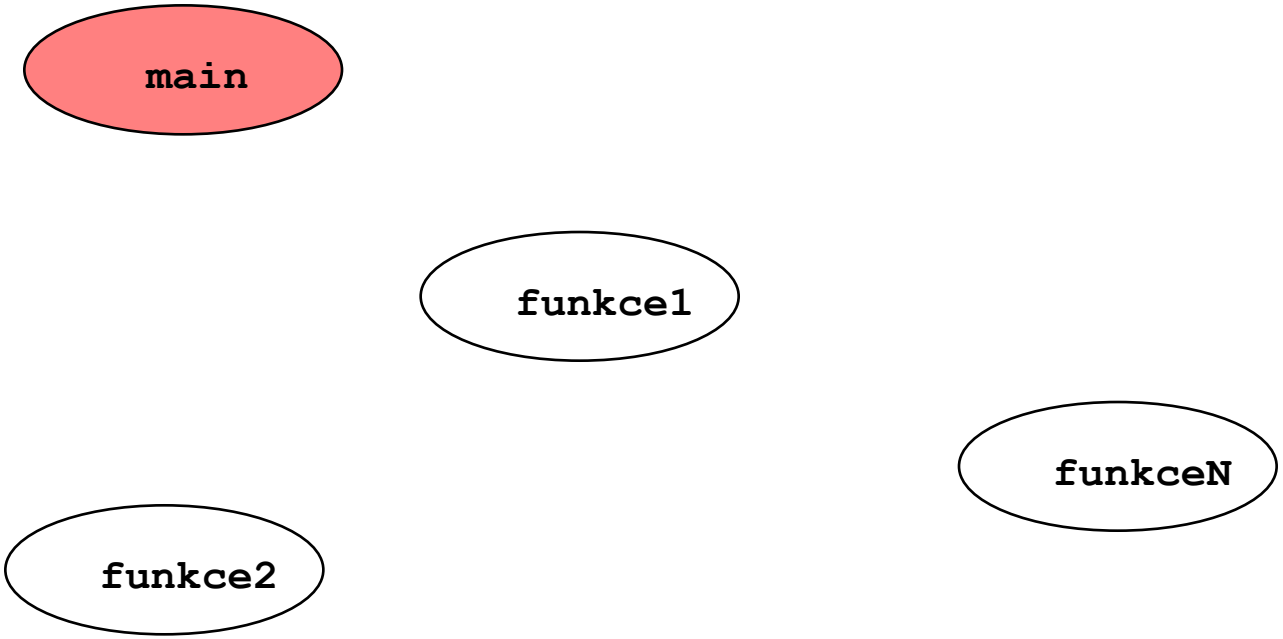
Základní charakteristika jazyků C a C++

- Procedurální vyšší programovací jazyky.
- Některé rysy nižších jazyků (asembleru – zejména C).
- Jednoduché (skalární) datové typy: celočíselné, racionální, ukazatelé.
- Strukturované datové typy: pole, řetězce, struktury, třídy.
- Bohaté operační struktury: aritmetické, logické, bitové, přiřazení.
- Základní řídicí struktury: podmíněné příkazy, přepínače, cykly, bloky, funkce.
- Vstup/výstup a ovládání dynamické volné paměti není součástí jazyka, ale standardních knihoven.
- Stručná (někdy nepřehledná) syntaxe.
- Program se skládá z modulů nezávisle překládaných, sestavených do řešení.
- Modul (kompilační jednotka) = zdrojový soubor obsahující deklarace a definice.
- Vstupním bodem programu je funkce **main.**

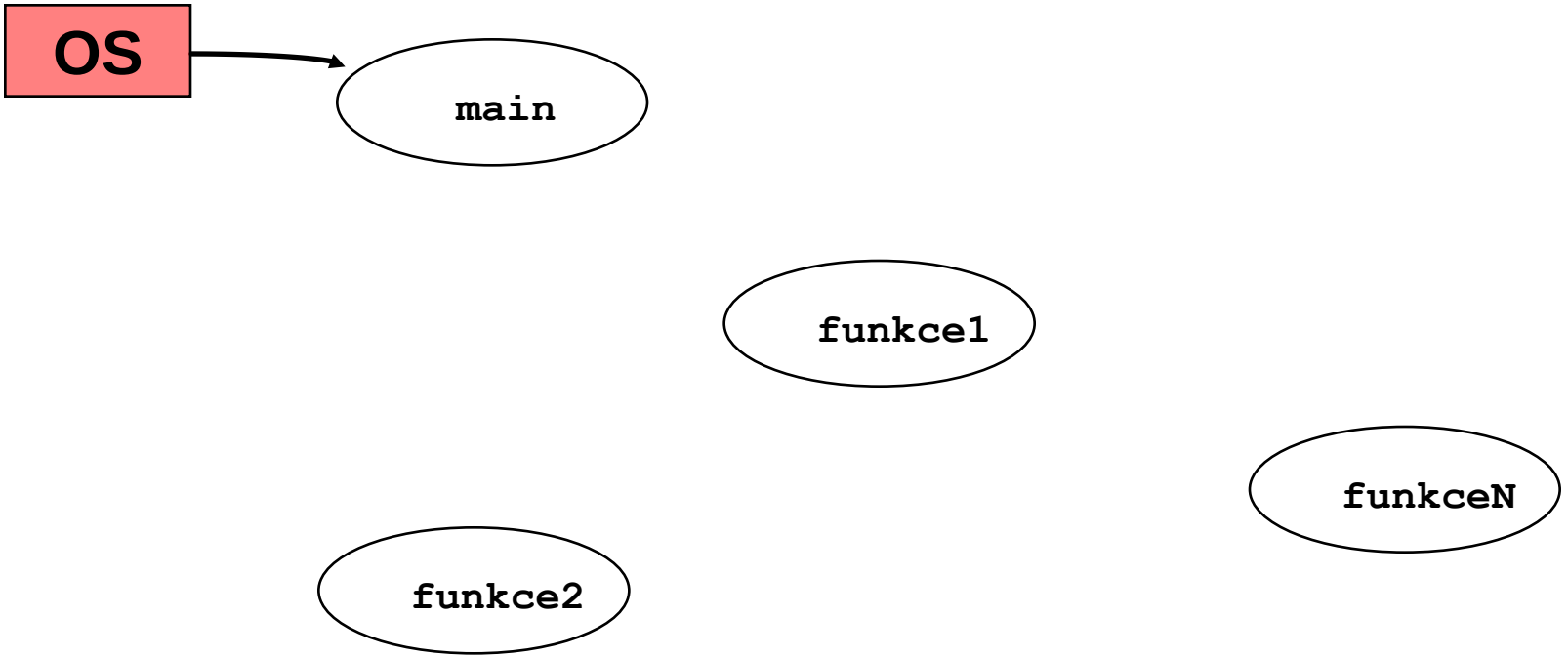
Program v C i C++ je sada funkcí



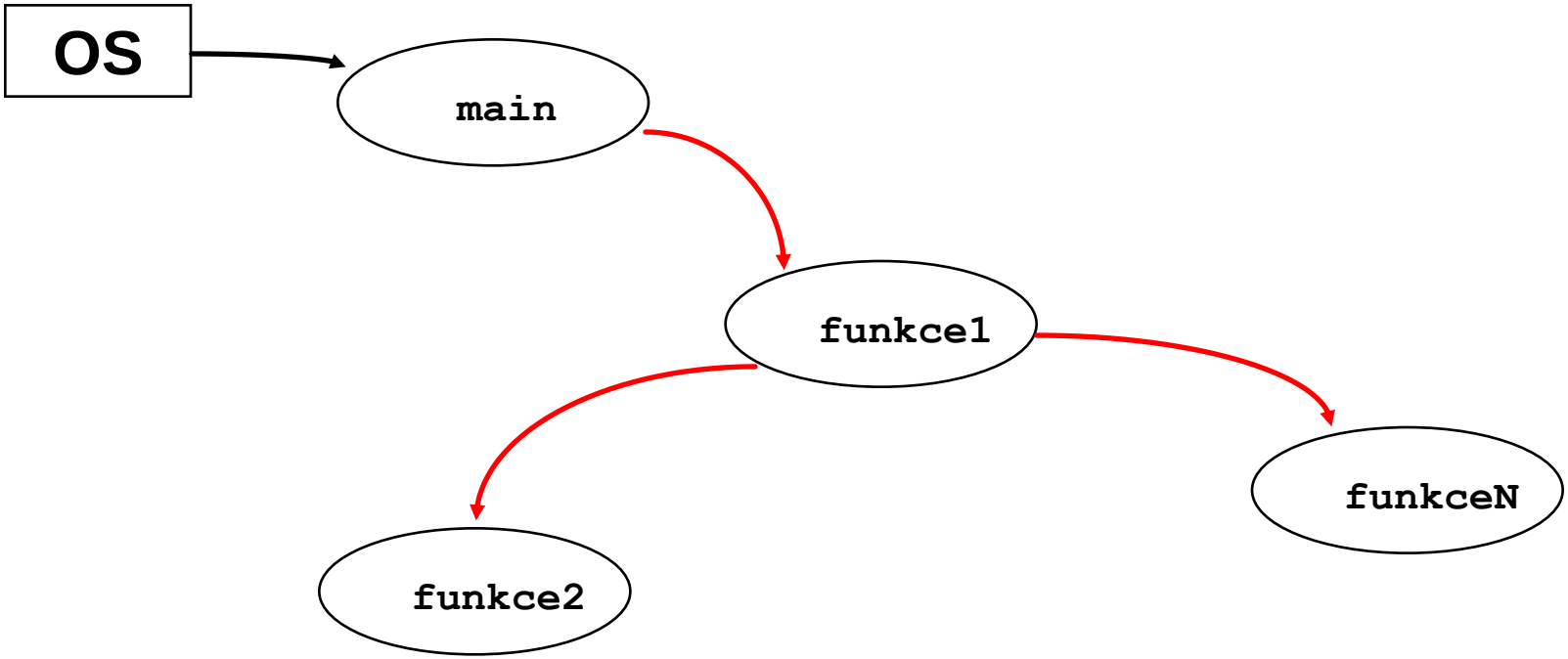
Jedna z nich se jmenuje `main` (nebo obdobně)



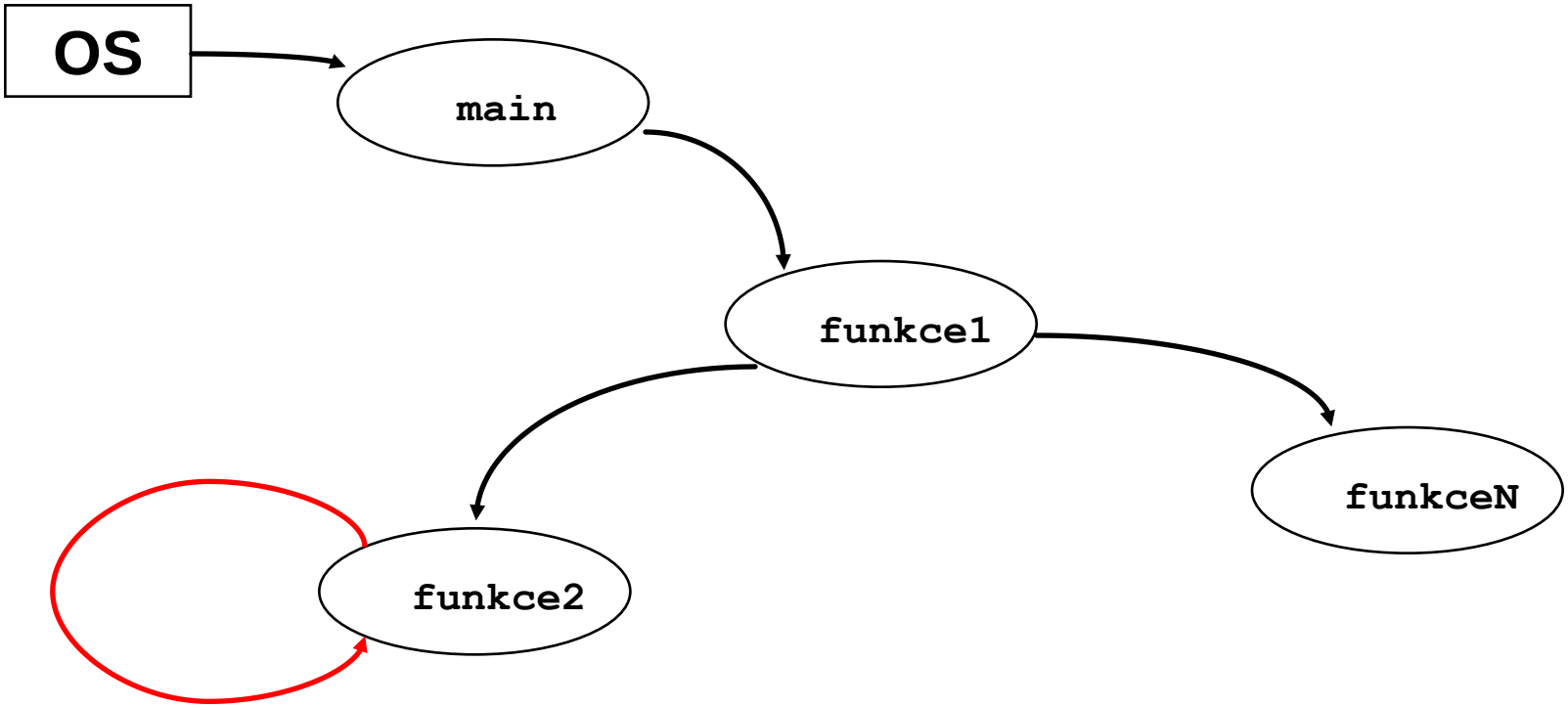
Funkci `main` spustí OS (shell)



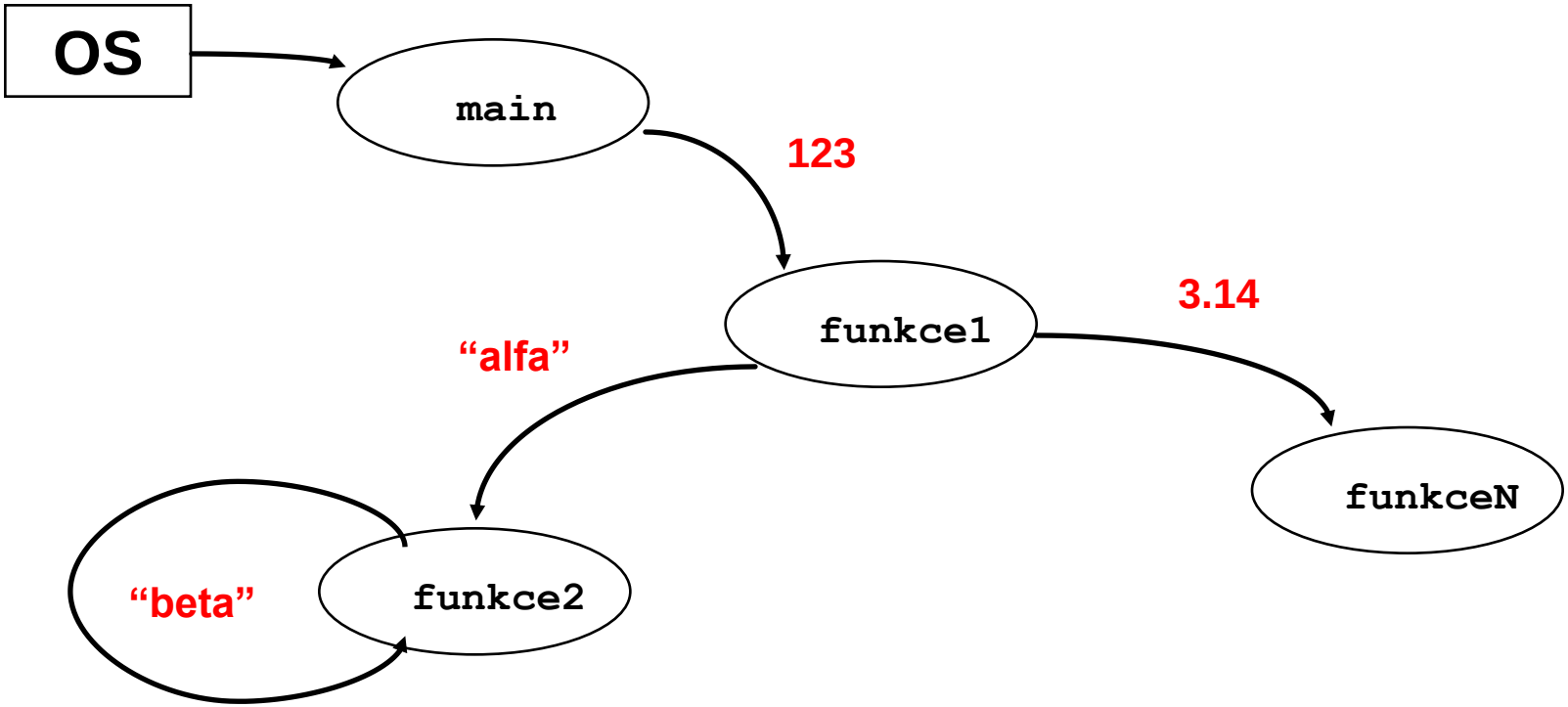
Funkce spolu komunikují (volají se)



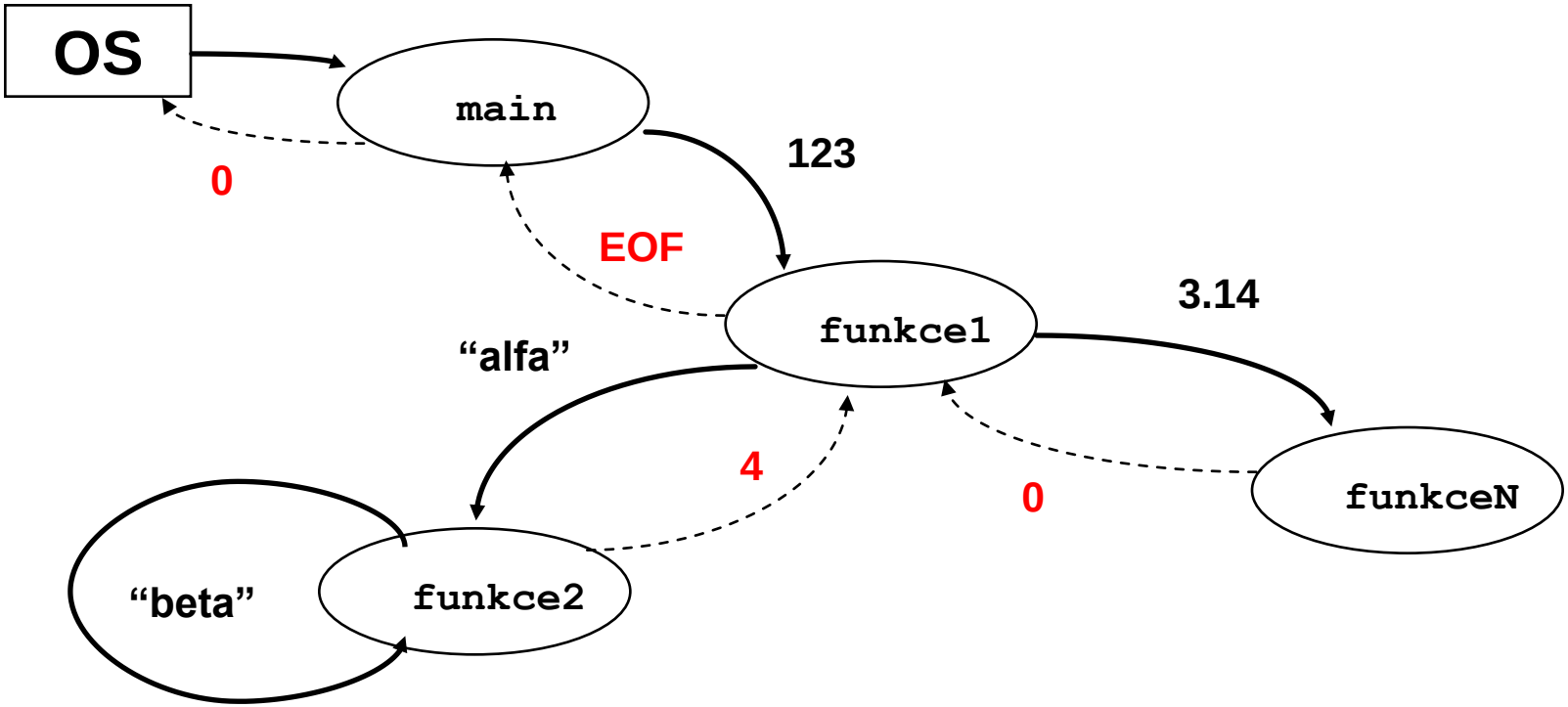
Případně i rekurzivně



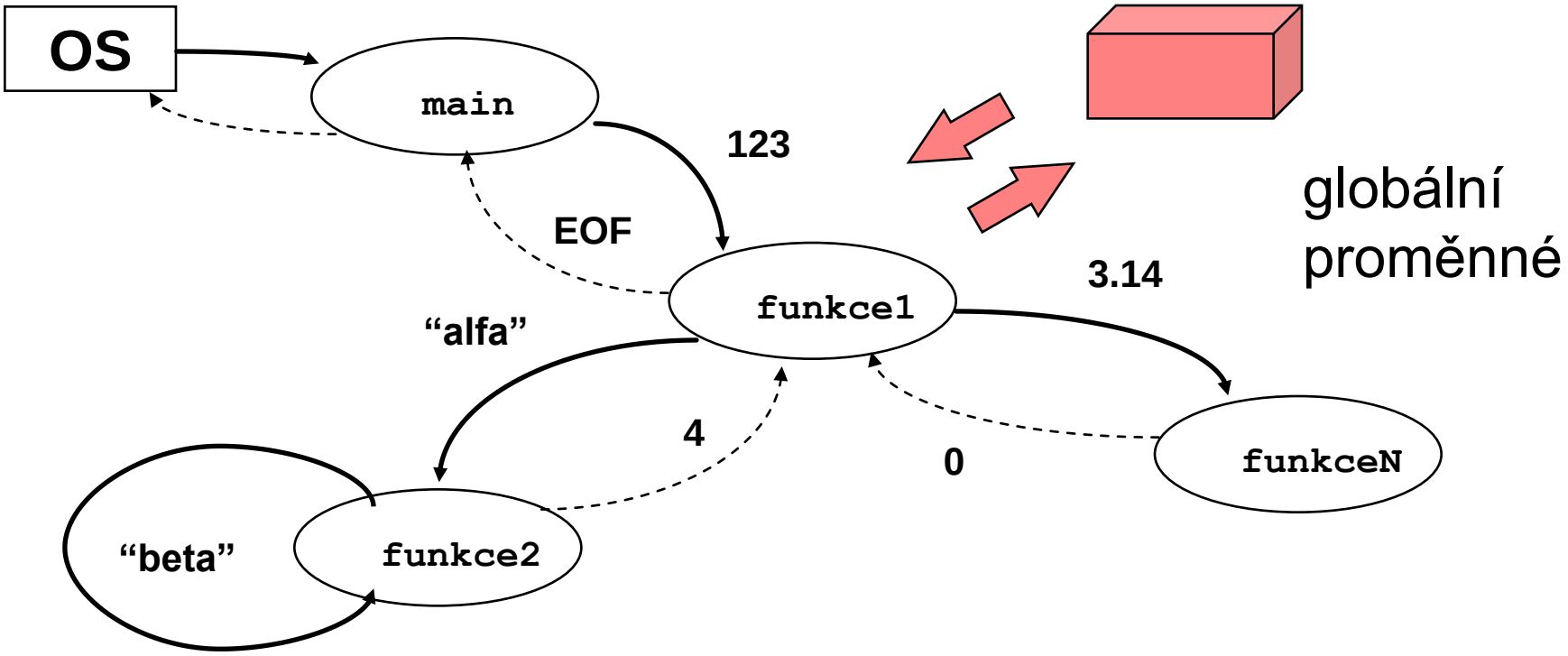
Předávají si parametry



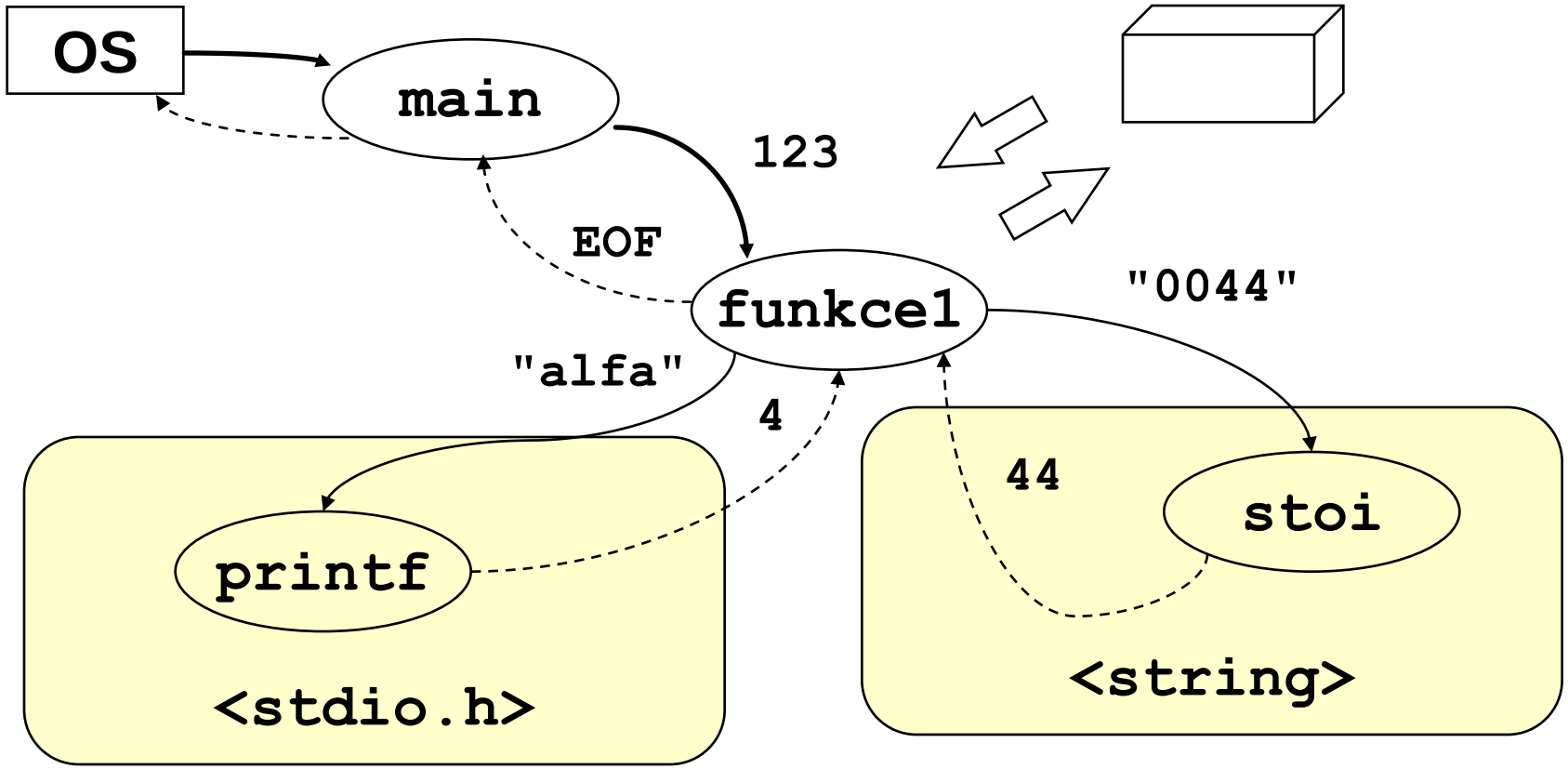
A vracejí výsledky



Navíc existuje globální paměť



Některé funkce jsou připraveny v knihovnách funkcí



Vstup a výstup v C++

- Každý program se chápe jako filtr – filtruje vstupní data na výstupní.
- Každý program proto má standardní vstup (cin – console input), standardní výstup (cout – console output) a výstup pro hlášení chyb (cerr – console errors).
- Vstup pomocí >>

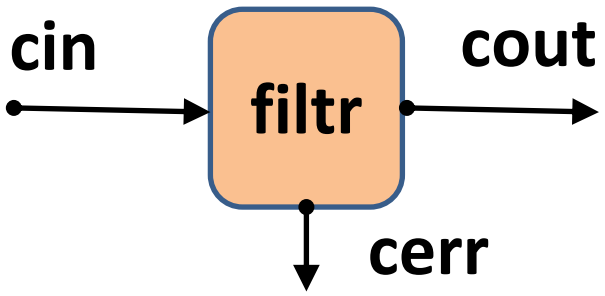
```
std::cin >> x;
```

- Výstup pomocí <<

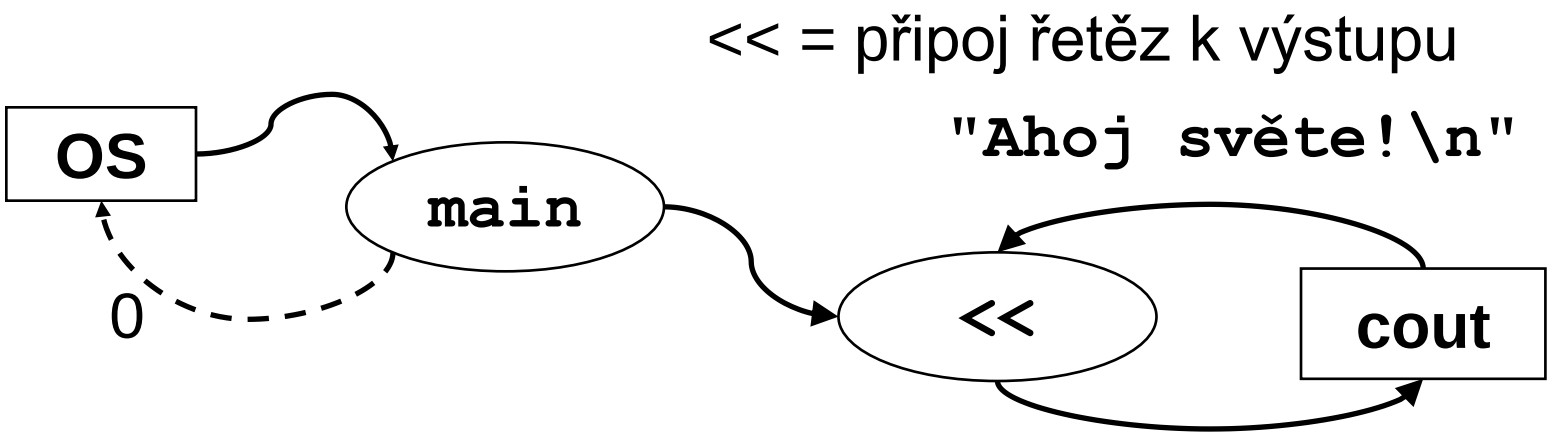
```
std::cout << x;
```

- Konec řádku: "\n" nebo endl

```
std::cout << std::endl;
std::cout << "\n";
```



Příklad “Ahoj světe!” v C++




```
#include <iostream>

int main()
{
        std::cout << "Ahoj světe!\n";
        return 0;
}
```

Vstup a výstup v C

- Každý program se chápe jako filtr – filtruje vstupní data na výstupní.
- Každý program proto má standardní vstup (stdin – standard input), standardní výstup (stdout – standard output) a výstup pro hlášení chyb (stderr – standard errors).
- Vstup pomocí scanf

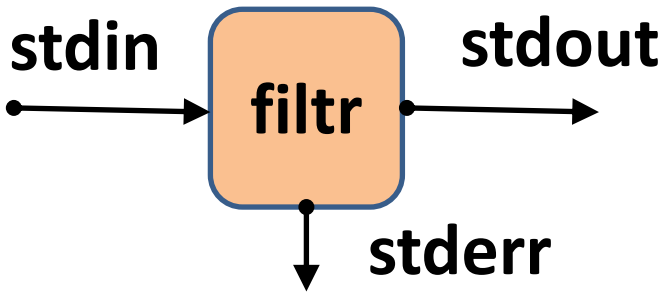
```
scanf("%d",&x);
```

- Výstup pomocí printf

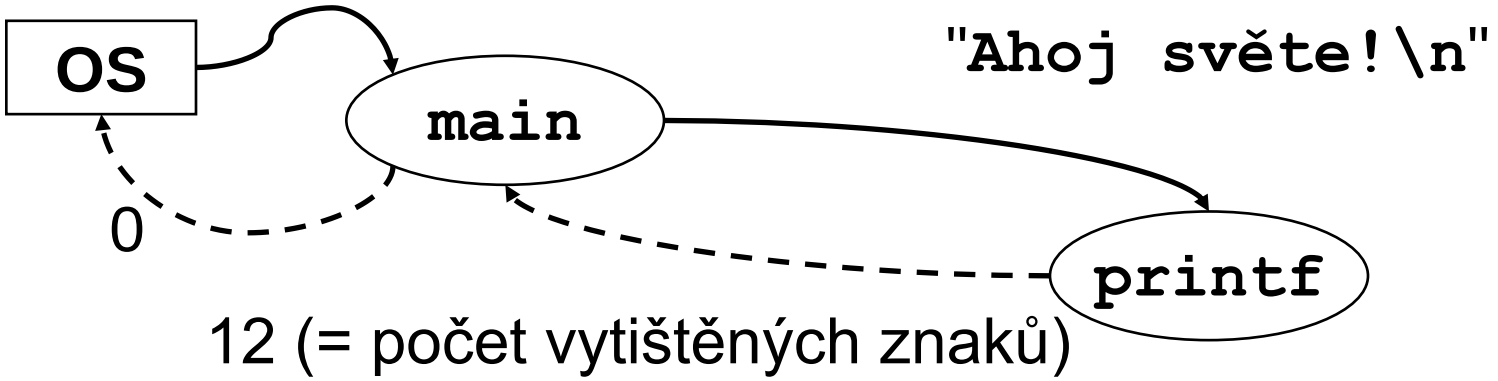
```
printf("%d",x);
```

- Konec řádku: "\n"

```
printf("\n");
```



Příklad “Ahoj světe!” v C




```
#include <stdio.h>

int main()
{
        printf("Ahoj světe!\n");
        return 0;
}
```

Preprocesor

- Povinná součást prostředí C a C++
- Provádí textové úpravy před vlastním překladem
- Řídí se příkazy pro preprocesor – **direktivami**
- Formát direktivy:

  #direktiva parametry

- Příklady direktiv:
  - Vložení textu

    #include <soubor>

    #include “soubor”
  - Definice makra

    #define jméno text

    #define jméno(parametry) text
  - Podmíněný překlad, parametry překladu (později)

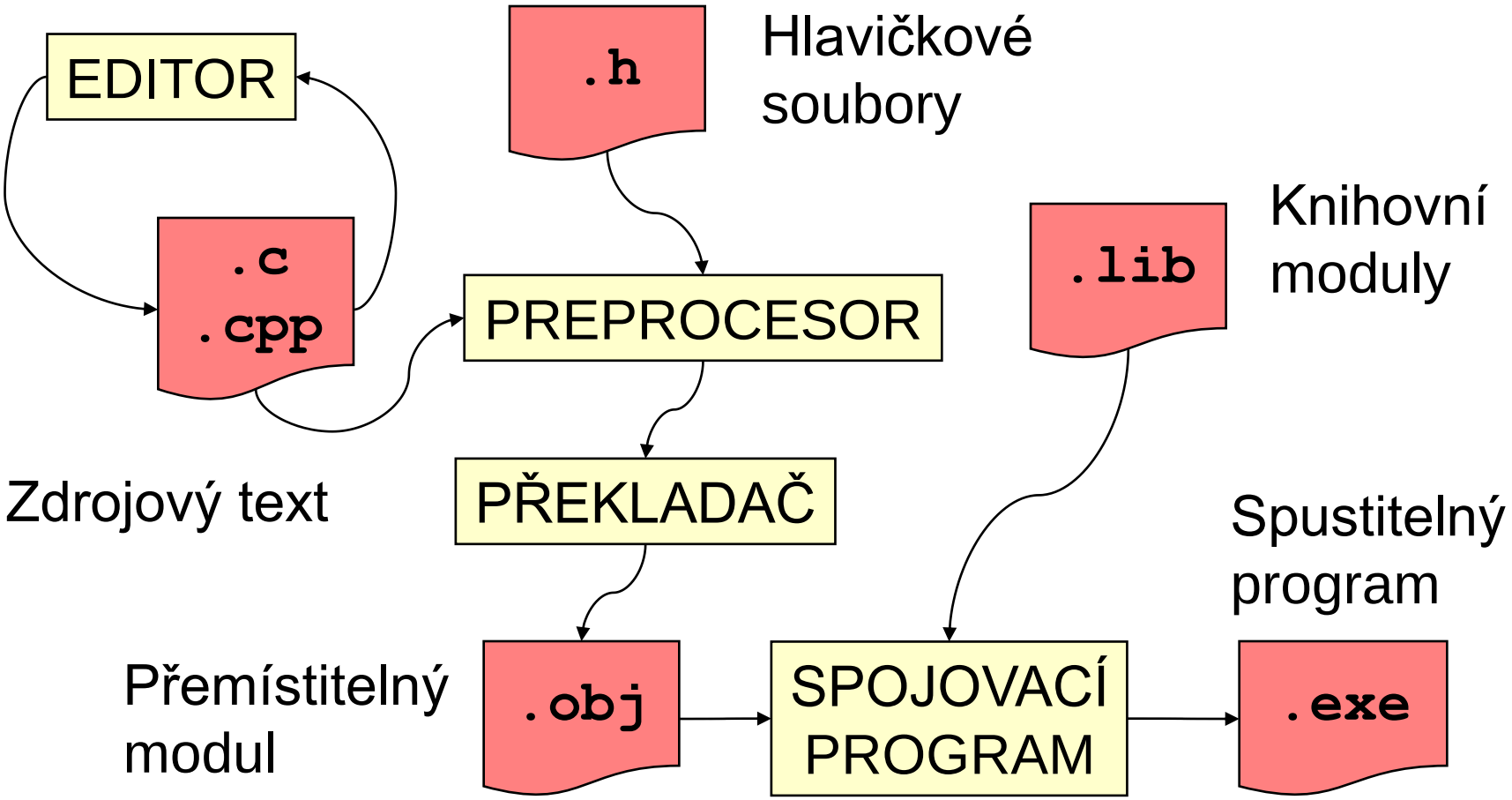
Preprocesor (pokrač.)

Příklad: *Vložení souboru*

```
#include <iostream>
...
std::cout << "Ahoj světe!\n";
```

objekt **`std::cout`** je součástí knihovny **`iostream`** (standardní vstup a výstup) a je popsán (deklarován) v hlavičce **`<iostream>`**

Fáze překladu



Ukázka programu v C++

```
/* Ahoj světe */
```

komentář

```
#include <iostream>
```

použití knihovny **`iostream`**

```
int main()
{
    std::cout << "Ahoj světe!\n";
    return 0;
}
```

funkce **`main`** používá operátor **`<<`** a objekt **`cout`**

Formát zápisu programu v C/C++ není předepsán

```
#include <stdio.h>
int main(int t, int _, char *a){return!0<t?t<3?main(-79,-13,a+main(-87,1-_,
main(-86,0,a+1)+a)):1,t<_?main(t+1,_,a):3,main(-94,-27+t,a)&&t==2?_<13?
main(2,_+1,"%s %d %d\n"):9:16:t<0?t<-72?main(_,t,
"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,/#{l,+,/n{n+,/+#n+,/#\
;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K w'K:'+}e#';dq#'l \
q#'+d'K#!/+k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n'){)#}w'){){nl]'/+#n';d}rw' i;# \
){nl]!/n{n#'; r{#w'r nc{nl]'/#{l,+'K {rw' iK{;[{nl]'/w#q#n'wk nw' \
iwk{KK{nl]!/w{%'l##w#' i; :{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \
;;{nl'-{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;#'rdq#w! nr'/ ') }+}{rl#'{n' ')# \
}'+}##(!!/")
:t<-50?_==*a?putchar(31[a]):main(-65,_,a+1):main((*a=='/')+t,_,a+1)
:0<t?main(2,2,"%s"):*a=='/'||main(0,main(-61,*a,
"!ek;dc i@bK'(q)-[w]*%n+r3#l,{}:\nuwloca-O;m .vpbks,fxntdCeghiry"),a+1);}
```

Zdroj: Wikipedia, International Obfuscated C Code Contest

Výstup tohoto programu

On the first day of Christmas my true love gave to me a partridge in a pear tree.

On the second day of Christmas my true love gave to me two turtle doves and a partridge in a pear tree.

On the third day of Christmas my true love gave to me three French hens, two turtle doves and a partridge in a pear tree.

On the fourth day of Christmas my true love gave to me four calling birds, three French hens, two turtle doves and a partridge in a pear tree.

On the fifth day of Christmas my true love gave to me five gold rings; four calling birds, three French hens, two turtle doves and a partridge in a pear tree.

On the sixth day of Christmas my true love gave to me six geese a-laying, five gold rings; four calling birds, three French hens, two turtle doves and a partridge in a pear tree.

On the seventh day of Christmas my true love gave to me seven swans a-swimming, six geese a-laying, five gold rings; four calling birds, three French hens, two turtle doves and a partridge in a pear tree.

On the eighth day of Christmas my true love gave to me eight maids a-milking, seven swans a-swimming, six geese a-laying, five gold rings; four calling birds, three French hens, two turtle doves and a partridge in a pear tree.

On the ninth day of Christmas my true love gave to me nine ladies dancing, eight maids a-milking, seven swans a -swimming, six geese a-laying, five gold rings; four calling birds, three French hens, two turtle doves and a partridge in a pear tree.

On the tenth day of Christmas my true love gave to me ten lords a-lea ping, nine ladies dancing, eight maids a-milking, seven swans a-swimming, six geese a-laying, five gold rings; four calling birds, three French hens, two turtle doves and a partridge in a pear tree.

On the eleventh day of Christmas my true love gave to me eleven pipers piping, ten lords a-lea ping, nine ladies dancing, eight maids a-milking, seven swans a-swimming, six geese a-laying, five gold rings; four calling birds, three French hens, two turtle doves and a partridge in a pear tree.

On the twelfth day of Christmas my true love gave to me twelve drummers drumming, eleven pipers piping, ten lords a-lea ping, nine ladies dancing, eight maids a-milking, seven swans a-swimming, six geese a-laying, five gold rings;

four calling birds, three French hens, two turtle doves and a partridge in a pear tree.

Výstup tohoto programu (česky)

Na první den Vánoc mi moje pravá láska dala koroptev v hrušce (Ježíš Kristus).

Na druhý den Vánoc mi moje pravá láska dala dvě hrdličky (Starý a Nový zákon) a koroptev v hrušce.

Na třetí den Vánoc mi moje pravá láska dala tři francouzské slepice (víra, naděje, láska), dvě hrdličky a koroptev v hrušce.

Na čtvrtý den Vánoc mi moje pravá láska dala čtyři volající ptáky (evangelisté), tři francouzské slepice, dvě hrdličky a koroptev v hrušce.

Na pátý den Vánoc mi moje pravá láska dala pět zlatých prstenů (5 knih Mojžíšových), čtyři volající ptáky, tři francouzské slepice, dvě hrdličky a koroptev v hrušce.

Šestý den Vánoc mi má pravá láska dala šest snášejících hus (šest dní stvoření světa), pět zlatých kroužků, čtyři volající ptáky, tři francouzské slepice, dvě hrdličky a koroptev v hrušce.

Sedmý den Vánoc mi moje pravá láska dala sedm plovoucích labutí (sedm darů Ducha svatého), šest snášejících hus, pět zlatých prstenů, čtyři volající ptáky, tři francouzské slepice, dvě hrdličky a koroptev v hrušce.

Osmého dne Vánoc mi moje pravá láska dala osm dojících služek (osm blahoslavenství), sedm plovoucích labutí, šest snášejících hus, pět zlatých prstenů, čtyři volající ptáky, tři francouzské slepice, dvě hrdličky a koroptev v hrušce.

V devátý den Vánoc mi moje pravá láska dala devět tančících dam (ovoce Ducha svatého), osm dojících služek, sedm plovoucích labutí, šest snášejících hus, pět zlatých kroužků, čtyři volající ptáky, tři francouzské slepice, dvě hrdličky a koroptev v hrušce.

Desátý den Vánoc mi má pravá láska dala deset skákajících pánů (Desatero přikázání), devět tančících dam, osm dojících služek, sedm plovoucích labutí, šest snášejících hus, pět zlatých kroužků, čtyři volající ptáky, tři francouzské slepice, dvě hrdličky a koroptev v hrušce.

Jedenáctého dne Vánoc mi moje pravá láska dala jedenáct dudajících dudáků (jedenáct věrných apoštolů), deset skákajících pánů, devět tančících dam, osm dojících služek, sedm plovoucích labutí, šest snášejících hus, pět zlatých kroužků, čtyři volající ptáky, tři francouzské slepice, dvě hrdličky a koroptev v hrušce.

Na dvanáctý den Vánoc mi moje pravá láska dala dvanáct bubnujících bubeníků (dvanáct bodů víry apoštolů), jedenáct dudajících dudáků, deset skákajících pánů, devět tančících dam, osm dojících služek, sedm plovoucích labutí, šest snášejících hus, pět zlaté prsteny, čtyři volající ptáky, tři francouzské slepice, dvě hrdličky a koroptev v hrušce.

Struktura programu v C++

- Program v C++ realizující řešení problému je sada definic tříd a funkcí, jedna z funkcí se jmenuje **main** (nebo podobně – to stanoví konkrétní prostředí) a ta představuje hlavní program, který se spustí. Funkce **main** pak případně volá jiné funkce, či metody realizované ve třídách.
- Definice funkce **main** má např. tvar:

```
int main() {
  int hodn = 0;
  char volba;
  std::cout << "\nPriklad – tisk hodnoty do zadani nuly\n";
  do {
    std::cin.get(volba);
    std::cout << "Hodnota citace= " << hodn << "\n";
  } while (volba != '0');
  std::cout << "Konec\n";
}
```

Různé styly řešení problémů

- Zápis programu realizuje určitou funkci či algoritmus, které představují řešení určitého problému.
- K řešení problému můžeme dospět různými způsoby – styly.
- Dva základní styly, kterými se budeme zabývat, jsou strukturovaný styl a objektově-orientovaný styl.
- Každý z nich se hodí na řešení trochu jiných problémů.
- Strukturovaný styl se hodí pro řešení procesních problémů, kdy řešení skládáme z posloupností jednotlivých kroků.
- Objektově-orientovaný styl se hodí spíše pro řešení různých informačních, či návrhových systémů, které uvažujeme jako síť komunikujících objektů.

Výpočetní problém

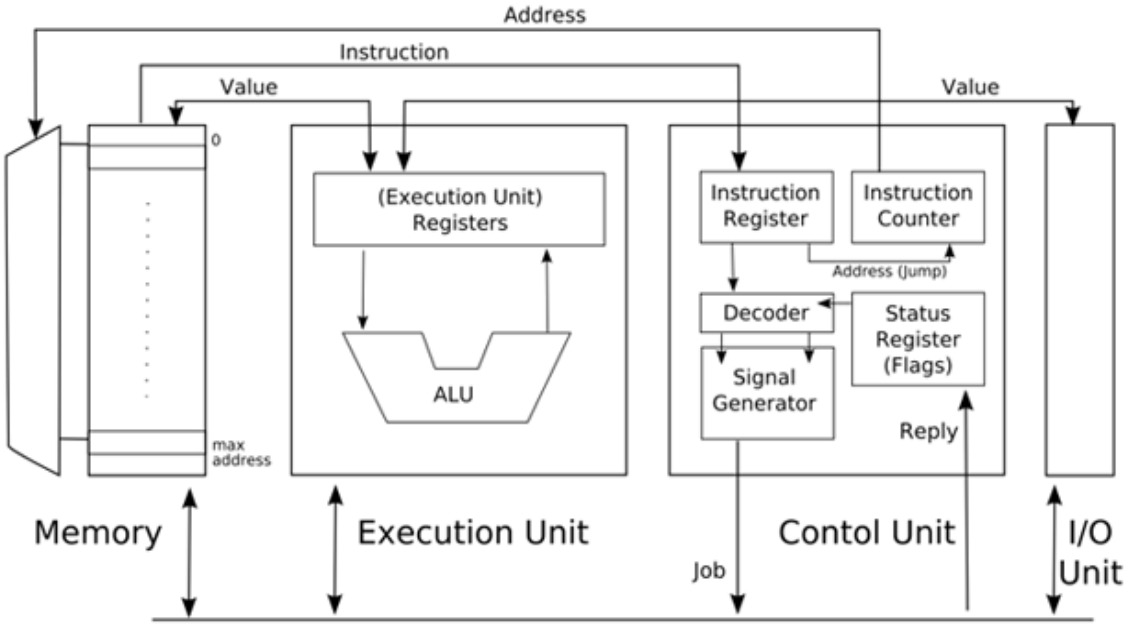
- Specifikace výpočetního problému má dvě části:
  - definici vstupu (vstupních dat) a
  - definici výstupu (výstupních dat).
- Instance výpočetního problému $\rightarrow$ nějaká konkrétní přípustná vstupní data. Obecně může k jedné instanci problému existovat několik správných výstupů (obecně se jedná o výpočet, představující relaci).
- Příklad: problém řazení čísel (Sorting Problem for Numbers).
  - Vstup: Posloupnost *n* čísel *Inp* = $\langle a_1,\ldots,a_n \rangle$.
  - Výstup: Taková permutace *Out* = $\langle a'_1,\ldots,a'_n \rangle$ stejných čísel z *Inp*, pro kterou platí, že: $a'_1 \leq \ldots \leq a'_n$.
- Instance problému řazení je např. zadání: seřaďte vzestupně posloupnost *Inp* = $\langle 75, 11, 34, 176, 59, 6, 54 \rangle$. Správným výstupem řazení je pak posloupnost *Out* = $\langle 6, 11, 34, 54, 59, 75, 176 \rangle$.
- Pozn.: Zde je výstup určen jednoznačně, ale obecně to neplatí.

Výpočetní stroj

John von Neumann

Von Neumannova architektura (1946):

- primitivní paměť (obsahuje data i program)
- sekvenční řízení výpočtu (čítač adres)



Algoritmus

- Neformálně: algoritmus je jakýkoli dostatečně přesný popis postupu výpočtu.
- Dostatečně přesný znamená:
  - v každém okamžiku postupu víme, který krok bude následovat (postup je deterministický),
  - pro libovolná vstupní data postup vždy skončí (je konečný) a
  - lze jej spustit pro všechny instance problému (je hromadný).
- Mezi algoritmy a výpočetními problémy nemusí být vztah jedna ku jedné – stejný problém lze zpravidla řešit různými algoritmy.
- Algoritmus řešící nějaký výpočetní problém je správný (korektní), pokud pro každou vstupní instanci skončí se správným výstupem. Správnost algoritmu se zpravidla musí dokázat (proč?).
- Algoritmus není program.
  - Program může být realizací algoritmu v určitém výpočetním prostředí.
  - Program nemusí být realizací algoritmu (nemusí být např. konečný).
- Různé algoritmy pro stejný problém mohou mít různou složitost.

Výpočetní problém - příklad

- **Úloha**: najděte největšího společného dělitele (NSD) dvou přirozených čísel (víme, co je největší společný dělitel dvou přirozených čísel?)
- Řešení:
  - Popišme postup tak, aby byl použitelný pro dvě libovolná přirozená čísla:
    - označme zadaná čísla x a y a menší z nich d
    - není-li d společným dělitelem x a y, pak d zmenšíme o 1, test opakujeme a skončíme, až d bude společným dělitelem x a y
- Poznámka:
  - Význam symbolů x, y a d použitých v algoritmu:
    - jsou to proměnné (paměťová místa), ve kterých je uložena nějaká hodnota, která se může v průběhu výpočtu měnit.

Algoritmus pro největšího spol.dělitele

Úloha: najděte největšího společného dělitele dvou přirozených čísel

Přesnější popis:

- **Vstup**: přirozená čísla x a y
- **Výstup**: NSD(x,y)
- **Postup**:
  1. Je-li x<y, pak d má hodnotu x, jinak d má hodnotu y
  2. Pokud je d dělitelem x nebo je d dělitelem y, jdi na krok 5
  3. Zmenši d o 1
  4. Jdi na krok 2
  5. Výsledkem je hodnota d

- Sestavili jsme algoritmus pro výpočet největšího společného dělitele dvou přirozených čísel.

Zápis algoritmu NSD v C++

```
#include <iostream>

int main() {
   int x,y;
   std::cout << "Zadej dve cisla: "; std::cin >> x >> y;

   int d;
   if (x < y) d = x; else d = y;

   while (!((x % d == 0) && (y % d == 0))) {
      d -= 1;
   };

   std::cout << "NSD(" << x << "," << y << ") = " << d << "\n";
};
```

Elementární algoritmické konstrukty

Posloupnost (sekvence):

- krok v algoritmu, který je považován za jednotku (nemusí být elementární operací) a je proveden pouze jednou,
- v případě více posloupností je jasně dáno pořadí jejich zpracování.

Větvení, výběr (selekce):

- podmíněný výběr průběhu algoritmu na základě vyhodnocení výběrové složky.

Cyklus (iterace):

- opakování části algoritmu (těla cyklu), které je prováděno dokud je/není splněna podmínka opakování.

Dílčí algoritmus (podprogram, funkce, metoda):

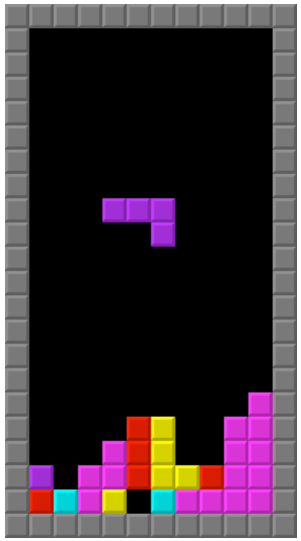
- část algoritmu, kterou lze použít opakovaně.

Svět objektů

- Objektově Orientované Programování (OOP):
  - Vychází z intuitivního chápání světa.
  - Základním konceptem jsou objekty.
  - Objekty mají určité *vlastnosti (properties)* a *schopnosti (capabilities)*.
- Objekty jsou stavební bloky programu:
  - Program je tedy soubor interagujících objektů.
- Objekty mohou modelovat **reálné věci**:
  - auto, hrnek, židle, zeměkoule.
- Objekty mohou modelovat **koncepty**:
  - schůzka, shromáždění, telefonát.
- Objekty mohou modelovat **procesy**:
  - hledání cesty bludištěm, seřazení balíčku karet.

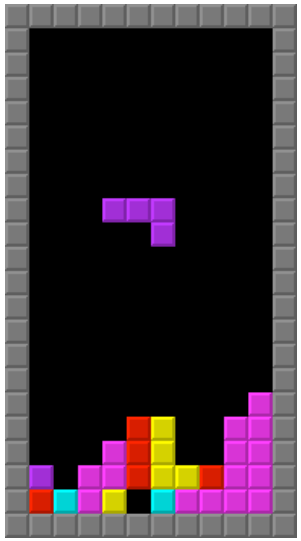
Příklad - Tetris

- Jaké jsou herní objekty?
- Co musí tyto objekty umět vykonat?
- Jaké vlastnosti tyto objekty mají?



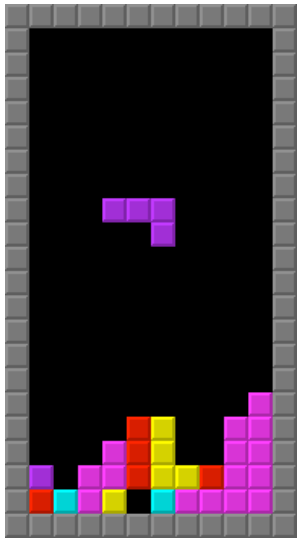
Příklad - Tetris

- Jaké jsou herní objekty?
  - např. dílky a hrací plán
- Co musí tyto objekty umět vykonat?
- Jaké vlastnosti tyto objekty mají?



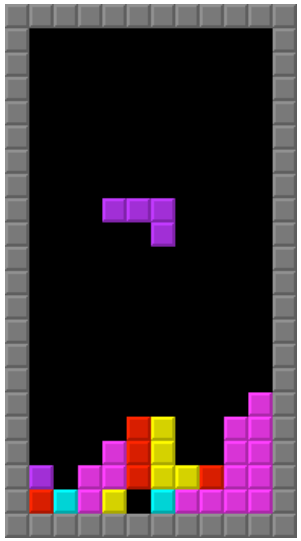
Příklad - Tetris

- Jaké jsou herní objekty?
  - např. dílky a hrací deska.
- Co musí tyto objekty umět vykonat?
  - dílek:
    - být vytvořen, spadnout, zastavit se, otočit se,
  - hrací deska:
    - být vytvořena, odstranit řádek,
    - zkontrolovat konec hry.
- Jaké vlastnosti tyto objekty mají?



Příklad - Tetris

- Jaké jsou herní objekty?
- Co musí tyto objekty umět vykonat?
- Jaké vlastnosti tyto objekty mají?
  - dílek
    - orientace, pozice, tvar, barva,
  - hrací deska
    - velikost, počet řádek.



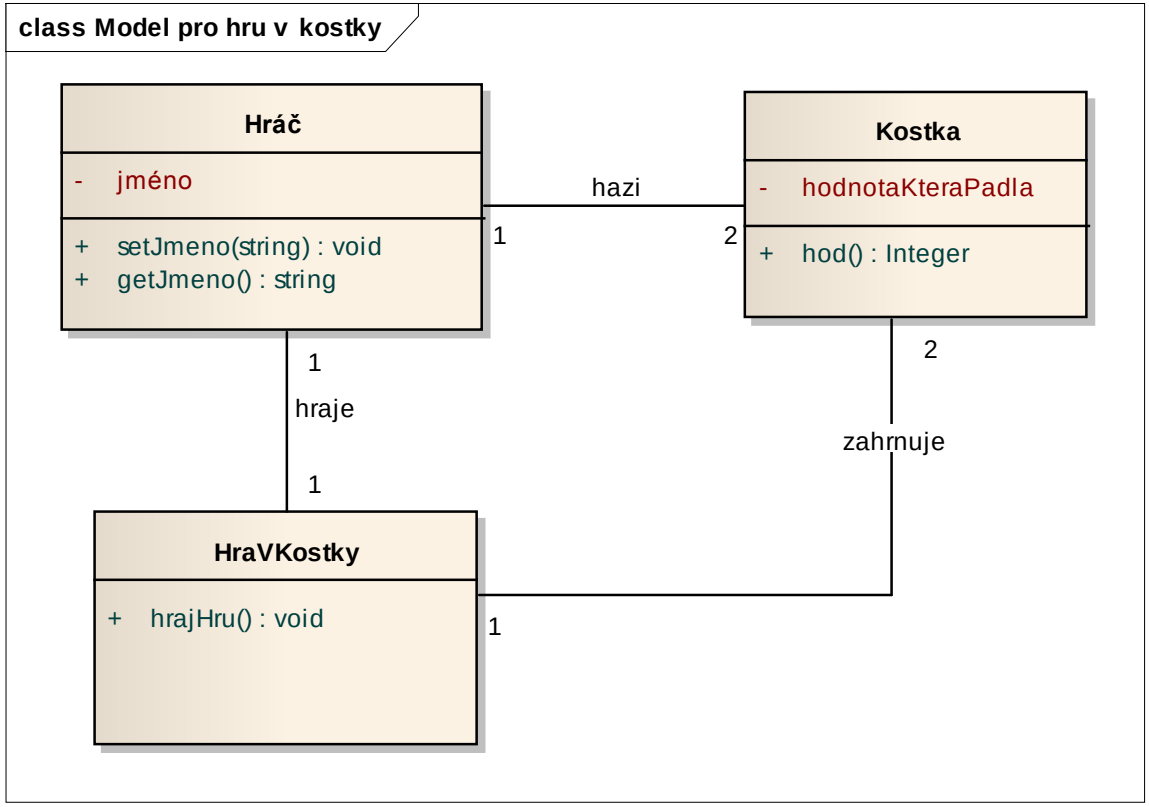
Stručný úvod do OO stylu

- Objektový model výpočtu si představuje řešení jako sadu objektů, které si posílají zprávy – požadují od jiných objektů vykonání nějaké akce, kterou samy neumí.
- Objekty jsou instance tříd, které shrnují jejich společné vlastnosti a definují jejich funkčnost – metody, které lze nad objekty, příp. s objekty provádět.
- Příklad: Představme si objekt typu fronta zákazníků – je to instance třídy FRONTA. Třída FRONTA umí provést metodu zarad_zákazníka, která do fronty zařadí dalšího zákazníka. Systém, který simuluje obsluhu zákazníků vytvoří nového zákazníka (objekt typu ZAKAZNIK) a nechá si ho zařadit do fronty zákazníků. Z fronty pak vybírá zákazníky, které bude obsluhovat.
- Abychom si mohli předvádět ukázky, řekneme si něco o struktuře objektového řešení a jeho zápisu v C++.

Příklad: Hra v kostky

- Uvažme příklad velmi jednoduché hry s kostkami – máme dvě kostky a házíme. Pokud nám vyjde součet v daném hodu 7, vyhráli jsme, jinak jsme naopak prohráli.
- Objektové řešení vychází z podstaty úlohy – pracujeme s objekty typu „kostka“. To jsou instance obecné třídy „Kostka“.
- Kostka umí metodu „hod“, která nastaví interní hodnotu kostky „hodnotaKteraPadla“ na náhodné číslo a vrátí toto číslo jako výsledek.
- Hru hrají hráči, kteří mají jméno – to je možno jim pomocí metod nastavit a číst (setJmeno/getJmeno).
- Hra spočívá v tom, že hráč hodí kostkami – to nám může zajistit metoda „hrajHru“.

Objektový model pro hru v kostky



Třída „Kostka“

```
class Kostka {
private:
   int hodnotaKteraPadla;
public:
   int hod() { return hodnotaKteraPadla = "náhodná hodnota"; };
};
```

```
#include <cstdlib>
class Kostka {
private:
   int hodnotaKteraPadla;
public:
   int hod() { return hodnotaKteraPadla = (std::rand() % 6) + 1;
   };
};
```

Třída Hráč

```
#include <string>

class Hrac {
private:
   std::string jmeno;
public:
   void setJmeno(std::string jm) { jmeno = jm; }
   std::string getJmeno() { return jmeno; }
};
```

Třída HraVKostky

```
#include <cstdlib>
#include <ctime>

class HraVKostky {
private:
   Kostka kostka1;
   Kostka kostka2;
public:
   HraVKostky() { std::srand((unsigned)std::time(NULL)); }
        /* nastavení náhodného generátoru */
   int hrajHru() {
        return kostka1.hod() + kostka2.hod();
   }
};
```

Hlavní program „main“ pro kostky

```
#include <iostream>
int main() {
   HraVKostky Hra;
   Hrac h;
   std::string jmeno;
   int vysledek;
   char volba;
   std::cout << "Hra v kostky\n";
   std::cout << "Vase jmeno: "; std::cin >> jmeno; h.setJmeno(jmeno);
   std::cout << "Nazdar " << h.getJmeno() << "\n";
   do {
        vysledek = Hra.hrajHru();
        if (vysledek == 7) std::cout << "Vyhral jsi!!!\n";
        else std::cout << "Prohral jsi, sorry, soucet je: " << vysledek
   << "\n";
        std::cout << "Pokracovat?: "; std::cin >> volba;
   } while (volba != 'n');
}
```

Hlavičkové soubory a MS VS C++

```
#ifndef __hlavicka_h
#define __hlavicka_h
class Test {
  int a;
  int b;
};
#endif

#pragma once
class Test {
  int a;
  int b;
};
```

Vložení předkompilovaného hlavičkového souboru (v něm uloženy soubory, které se nemění často) => zrychlení překladu =>

```
*.pch
#include "stdafx.h"

// funkce main je ve VS _tmain
int _tmain(int argc, _TCHAR* argv[])
{
   printf("%s","text1");
   cout<<"text2";
   return 0;
}
```

The End