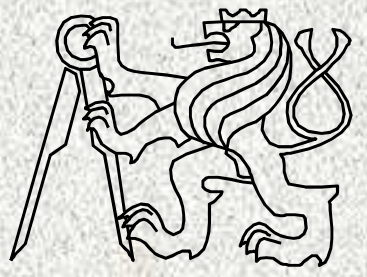
# Jazyk C - Základy



A0B36PR2-Programování 2

Fakulta elektrotechnická
České vysoké učení technické

# Cíl výkladu

- Součástí obsahu předmětu PR2 je výklad jazyka C na bázi znalosti jazyka Java
- Další výklad bude koncipován komparativně: v porovnání s možnostmi jazyka Java bude vysvětlován jazyk C

# C - Literatura

- **Herout,P.: Učebnice jazyka C. 3.vyd., Kopp, 2008**
- **Herout,P.: Učebnice jazyka C. 2.díl, Kopp, 2007**
- **Muldner, T: C for Java Programmers, Addison Wesley, 2000**

- Kernighan,B.-Ritchie,D.: The C Programming Language.2nded.1989
- Prinz,U.-Prinz,P.: C Pocket Reference. O'Reilly, 2002

- Johnson,M.P.: Programming Language C. 2003, www.columbia.edu/~mpj9/3101
- Alonso,G.: Programming in C. ETH Zurich, www.inf.ethz.ch/department/IS/iks

# Vývojová prostředí

Návody na cvičení

- Netbeans, http://www.cygwin.com/
- DEV-C++, **http://centrum.stahuj.cz**

# Obsah

- Literatura
- Historie
- Charakteristika
- C vs C++ vs  Java
- Stručně - C a Java - Co je stejné nebo podobné
- Stručně - C a Java - Co C nemá
- Stručně - C a Java - Co C nemá nebo je jinak
- Porovnání JAVA vs „C“
  - main(), „Nazdar Světe“
  - Proměnné, konstanty, přiřazení
  - Blok, místa pro deklaraci proměnných
  - Vstup, výstup
  - Řízení běhu programu (if, for, while, switch)
  - Operátory
  - Matematické funkce (knihovny)

# Použité barevné značení pro komparativní výklad

Rozlišení jazyka

Java

“C”

Srovnatelné vlastnosti

Java ← 1

“C” ← 1

Rozlišení stupně znalostí

Základní znalost

Požadovaná znalost

Doporučená znalost

Pro zájemce

Přehled

Pomůcka pro “C”

# C - Historie

- C – navržen 1972 (Dennis Ritchie, AT&T, Bell Laboratories, USA)

- C – ovlivněn:
  - ALGOL 60 (1960)
  - CPL (1963, Cambridge)
  - BCPL (1967, Martin Richard)
  - B (1970, Ken Thompson)

- 1973 - do C z asembleru přepsán operační systém UNIX
- 1978 - Kernighan & Ritchie – první standard C (K&R Standard)
- 1989 - ANSI C Standard (ANSI X3.159), známý jako ANSI C89
- 1990 - ISO C Standard (ISO/IEC 9899), vychází z C89
- 1995 - aktualizace C89 na ANSI C95
- 1999 - aktualizace C95 na ANSI C99

Pro zájemce

# C - Historie

- Skupina programovacích jazyků s částečně podobnou syntaxí:
  - C
    - 1972, Dennis Ritchie, Bell Laboratories
    - Procedurální jazyk
  - C++
    - 1986, Bjarne Stroustrup, Bell Laboratories
    - Objektová podpora
  - Java
    - 1995, James Gosling, Sun
    - Objektový
  - C#
    - 2002, Microsoft
    - Objektová podpora

Pro zájemce

# C - Charakteristika

- Univerzální programovací jazyk nižší až střední úrovně
  - *Strukturovaný* (funkce + data)
  - *Zdrojový kód přenositelný (portable), nutno ctít podmínky přenositelnosti*, překladač je závislý na platformě
  - *Rychlý, efektivní, kompaktní kód, mnohdy nepřehledný*
- Pružný a výkonný, ale *nestabilní*
- Podpora
  - konstrukcí jazyka vysoké úrovně (funkce, datové struktury)
  - operací blízkých assembleru (ukazatele, bitové operace,…)
    - *Slabá typová kontrola*
    - *Málo odolný programátorovým chybám*
- Dává velkou volnost programátorovi v zápisu programu
  - *Výhoda*: dobrý programátor vytvoří efektivní, rychlý a kompaktní kód
  - *Nevýhoda*: špatný nebo unavený programátor vytvoří nepřehledný program náchylný k chybám
- Použití: operační systémy, řídicí systémy, grafika, databáze, číslicové zpracování signálů (DSP),…

Základní znalost

# C vs. Java

Shrnutí:

- C
  - rychlý, kompaktní, málo bezpečný
  - kompilovaný kód
  - nutná vlastní správa paměti
- Java
  - bezpečná, elegantní, plně přenositelná, moderní
  - automatická správa paměti
  - interpretovaný kód (bytecode) s možnou částečnou podporou kompilovaných částí (JIT)

Poznámka:

- *C++*
  - *Rychlý, málo bezpečný, velmi složitý*
  - *Kompilovaný kód*

Základní znalost

# C vs. Java – oblasti použití

- C
  - programování ovladačů grafických, zvukových a dalších karet
  - programování vestavěných - embeded systémů
  - silná vazba na HW, využití jeho možností, špatná nebo žádná podpora národních zvyklostí
- Java
  - GUI
  - síťování po netu
  - složité aplikace vyžadující spolupráci mnoha programátorů
  - Nezávislá na OS i HW, standardizované typy, objektový přístup,

# C a Java - co je stejné nebo podobné

***v C je stejné (podobné)***

- Program začíná funkcí **`main(),`** je základní funkcí programu
- Stavba funkcí / metod,
  - Jméno funkce, formální parametry, návratová hodnota, vymezení těla funkce, vymezení bloku, vlastnosti lokálních proměnných (jsou v zásobníku), předávání primitivních typů parametrů hodnotou, **`return`**.
- Množina znaků pro konstrukci identifikátorů
- Primitivní typy proměnných se znaménkem (Java nezná proměnné bez znaménka)
  - **`char, short, int, long, float, double`**
- Aritmetické, logické, relační, bitové operátory
- Podmíněný příkaz: **`if() / if() else`**
- Příkazy cyklů: **`while() , do while(), for(;;), break, continue`**
- Programový přepínač: **`switch(), case, default, break`**

Základní znalost

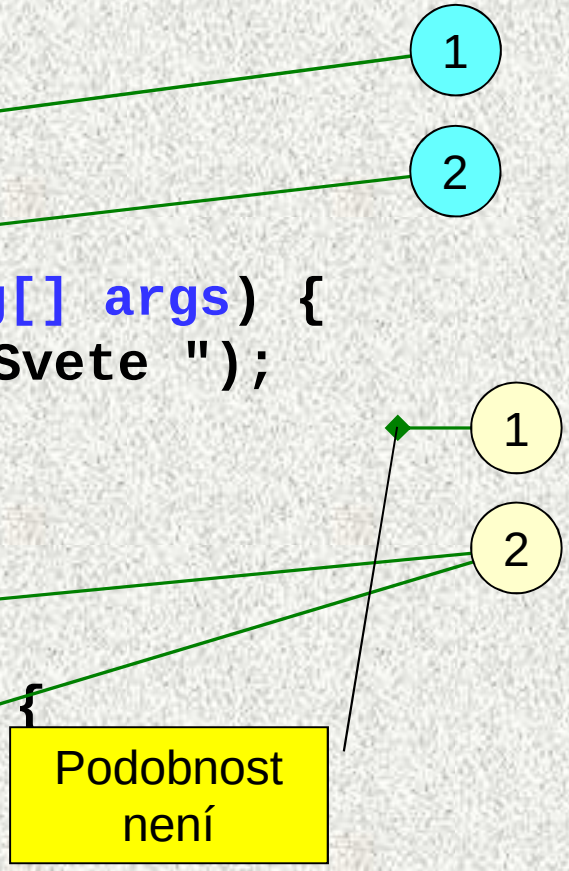
# JAVA vs C

- JAVA

```
public class PrvniProgram {
   public static void main(String[] args) {
      System.out.println("Nazdar Svete ");
   }
}
```

- C

```
int main (int argc, char** argv) {
    printf("Nazdar Svete \n");
    return (0);
}
```

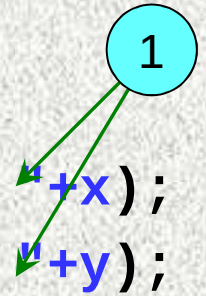



Základní znalost

# JAVA /podobné/ (1)

```
// Druhy program
public class DruhyProgram {

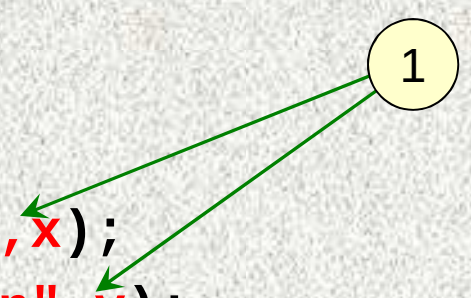
   public static void main(String[] args) {
     int x = 10, y;
     y = x + 20;
     System.out.println("Hodnota proměnné x je "+x);
     System.out.println("Hodnota proměnné y je "+y);
   }
 }
```




Základní znalost

# C /podobné/ (1)

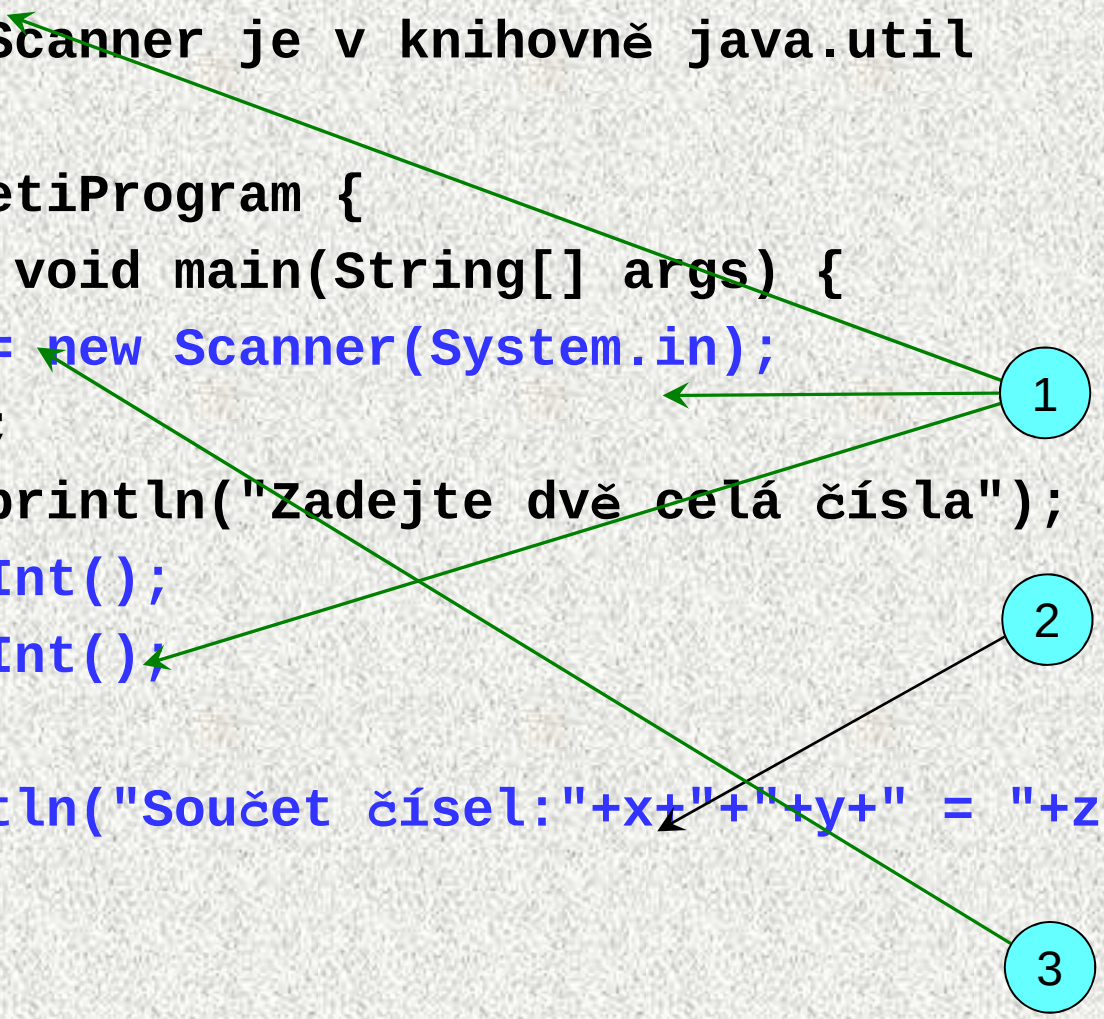
```
// Druhy Program
int main(int argc, char** argv) {
    int x = 10, y;
    y = x + 20;
    printf("Hodnota promenne x = %3d\n",x);
    printf("Hodnota promenne y = %3d\n\n",y);
    return (0);
}
```



# JAVA /jiné/ (2)

```
import java.util.*;
                // Scanner je v knihovně java.util

public class TretiProgram {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int x, y, z;
    System.out.println("Zadejte dvě celá čísla");
    x = sc.nextInt();
    y = sc.nextInt();
    z = x + y;
System.out.println("Součet čísel:"+x+"+"+y+" = "+z);
    }
}
```

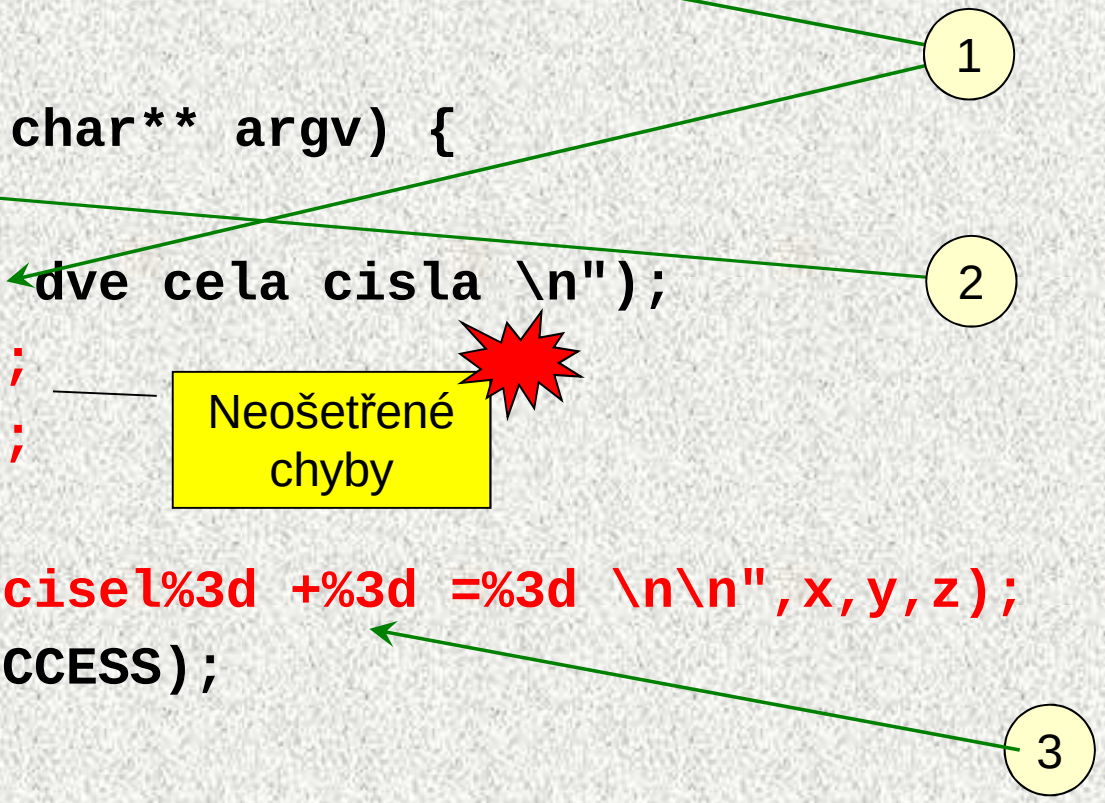



Základní znalost

# C /jiné/ (2)

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

// Tretí Program
int main(int argc, char** argv) {
    int x, y, z;
    printf("Zadejte dve cela cisla \n");
    scanf("%d", &x);
    scanf("%d", &y);
    z = x + y;
    printf("Soucet cisel%3d +%3d =%3d \n\n",x,y,z);
    return (EXIT_SUCCESS);
}
```




Základní znalost

# Načtení hodnoty ze stdin

- C

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

**&x**

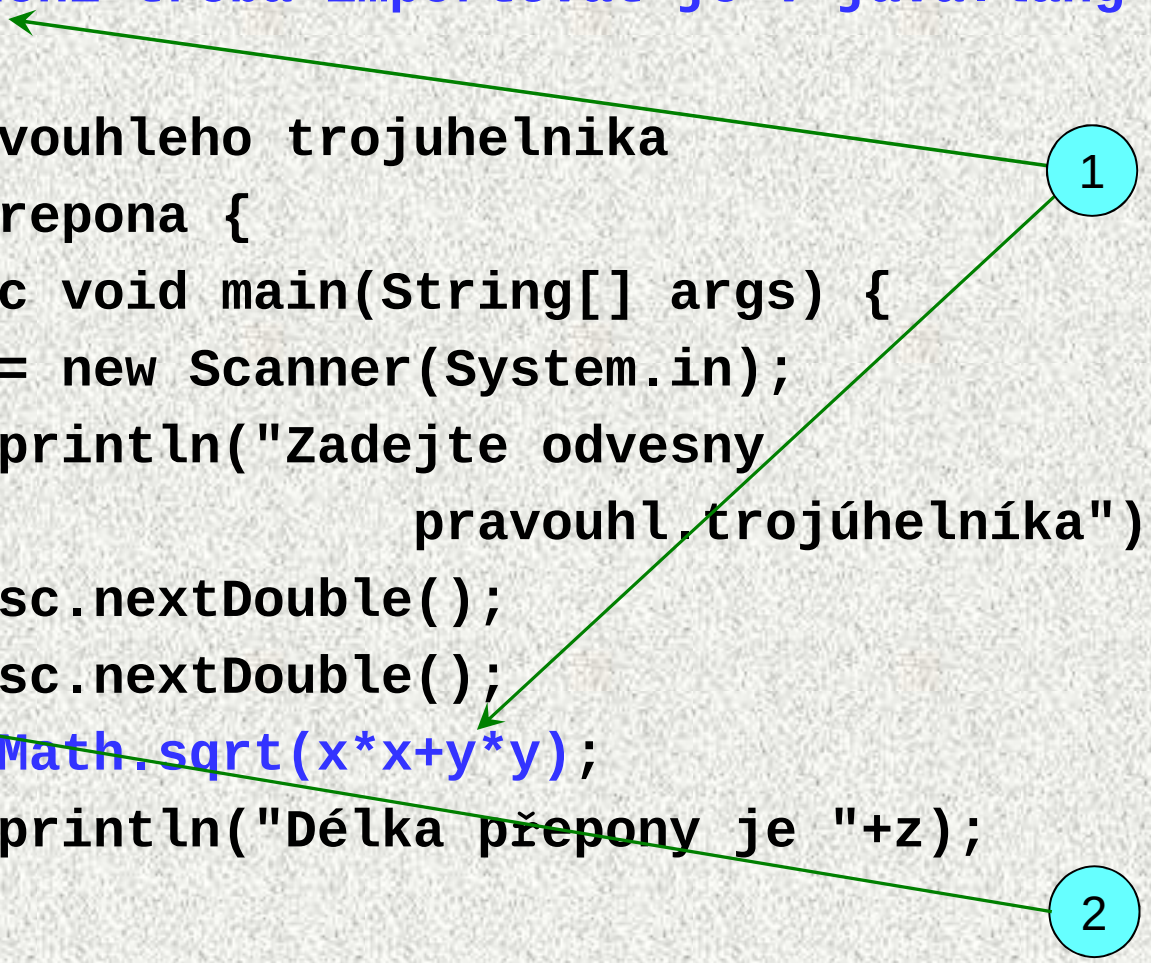
  - do funkce scanf vstupuje jako parametr adresa (resp. reference) paměťového místa, kam se má načtená hodnota uložit
  - jde o předání parametru odkazem, jde tedy o parametr, který může být využit pro vstup i výstup hodnoty
  - funkce v C může takto „vracet “ více hodnot
- Java
  - `x = sc.nextInt();`
  - parametry předáváme pouze hodnotou

Základní znalost

# JAVA /podobné/ (3)

```
import java.util.*;  // Scanner
// Tridu Math neni treba importovat je v java.lang

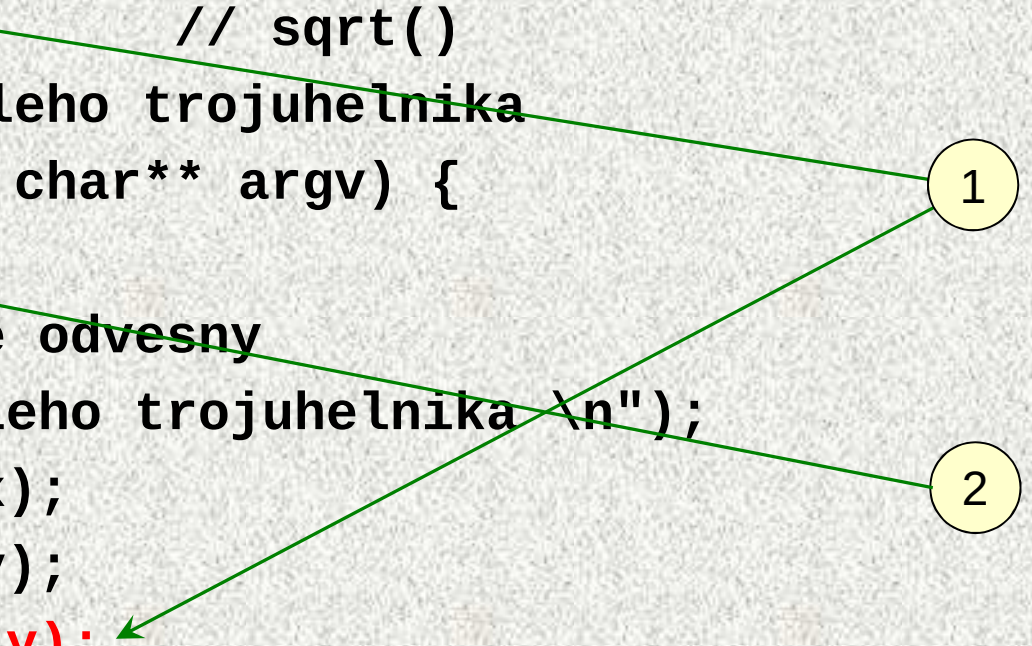
 // Prepona pravouhleho trojuhelnika
 public class Prepona {
   public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Zadejte odvesny
                                pravouhl.trojúhelníka");
    double x = sc.nextDouble();
    double y = sc.nextDouble();
    double z = Math.sqrt(x*x+y*y);
    System.out.println("Délka přepony je "+z);
   }
 }
```




Základní znalost

# C /podobné/ (3)

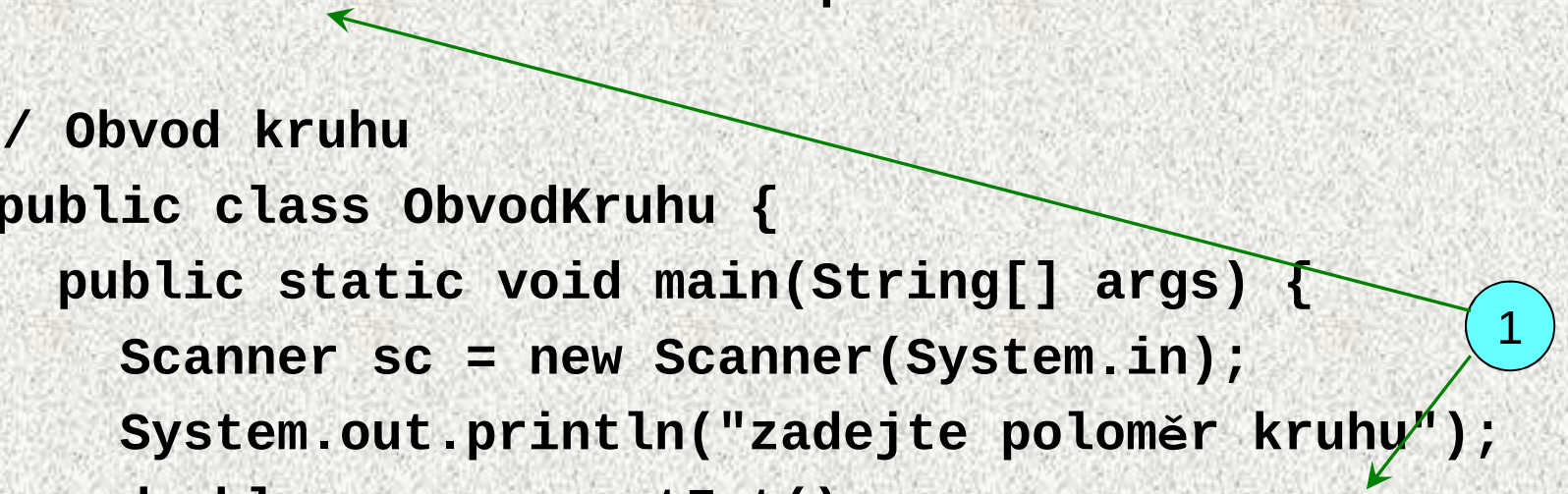
```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>          // sqrt()
// Prepona pravouhleho trojuhelnika
int main(int argc, char** argv) {
    double x, y, z;
    printf("Zadejte odvesny
           pravouhleho trojuhelnika \n");
    scanf("%lf", &x);
    scanf("%lf", &y);
    z = sqrt(x*x+y*y);
    printf("Delka prepony = %5.2f \n\n", z);
    return (EXIT_SUCCESS);
}
```



# JAVA /podobné/ (4)

```
import java.util.*;          // Scanner
// Tridu Math neni treba importovat
```

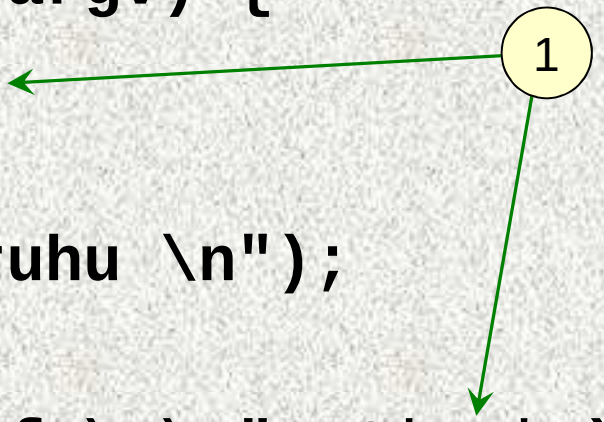
- ```
  // Obvod kruhu
   public class ObvodKruhu {
     public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);
       System.out.println("zadejte poloměr kruhu");
       double r = sc.nextInt();
   System.out.println("obvod kruhu je "+2*Math.PI*r);
     }
   }
  ```


  ```

# C /podobné/ (4)

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

// Obvod kruhu
int main(int argc, char** argv) {
const double PI = 3.1416;
double r;
printf("Zadejte polomer kruhu \n");
scanf("%lf", &r);
printf("Obvod kruhu = %5.2f \n\n", 2*PI*r);
printf("Obvod kruhu = %f \n\n", 2*PI*r);
return (EXIT_SUCCESS);
}
```




Základní znalost

# JAVA – if else /téměř shodné/ (5)

```
import java.util.*;
// Prestupny rok
public class Rok {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int rok;
    System.out.println("zadejte rok");
    rok = sc.nextInt();
    System.out.println("rok "+rok);
    if (rok > 1582){
       if ((rok%4==0 && rok%100!=0) || rok%400==0)
           printf(" je prestupny \n\n");
        else
           printf(" neni prestupny \n\n");
    }else{
       if((rok >= 8) && rok%4 == 0)
           printf(" je prestupny \n\n");
       else
          printf(" neni prestupny \n\n");
    }
  }
}
```


Základní znalost

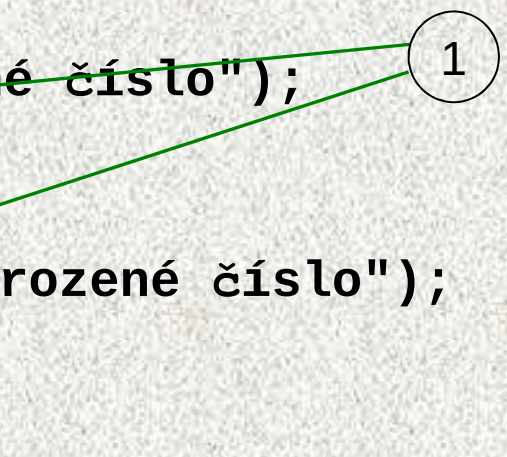
# C – if else /téměř shodné/ (5)

```
#include <stdio.h>
#include <stdlib.h>
// Prestupny rok
int main(int argc, char** argv) {
    int rok;
    printf("Zadejte rok = ");
    scanf("%d", &rok);
    printf("\nRok %4d",rok);
    if (rok > 1582){
       if ((rok%4==0 && rok%100!=0) || rok%400==0)
           printf(" je prestupny \n\n");
        else
           printf(" neni prestupny \n\n");
    }else{
        if((rok >= 8) && rok%4 == 0)
           printf(" je prestupny \n\n");
        else
          printf(" neni prestupny \n\n");
    }
return (EXIT_SUCCESS);
}
```


Základní znalost

# JAVA – while /shodné/ (6)

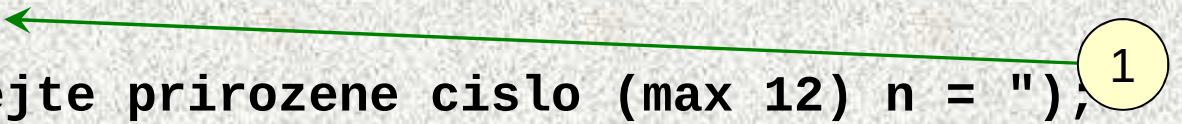
```
public class Faktorial {
   public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("zadejte přirozené číslo");
     int n = sc.nextInt();
     if (n < 1) {
       System.out.println(n + " není přirozené číslo");
       System.exit(0);
     }
     int i = 1;
     int f = 1;
     while (i < n) {
       i = i + 1;
       f = f * i;
     }
     System.out.println (n + "! = " + f);
   }
 }
```




Základní znalost

# C – while /shodné/ (6)

```
// Faktoriál
int main(int argc, char** argv) {
    int n, i, f;
    printf("Zadejte prirozene cislo (max 12) n = ");
    scanf("%d", &n);
    if(n < 1){
      printf("\n n = %d neni prirozene cislo \n\n", n);
     exit(EXIT_FAILURE);
    }
    i = 1;
    f = 1;
    while(i < n){
        i = i + 1;
        f = f * i;
    }
    printf("\n %3d! = %d \n\n", n, f);
    return (EXIT_SUCCESS);
}
```

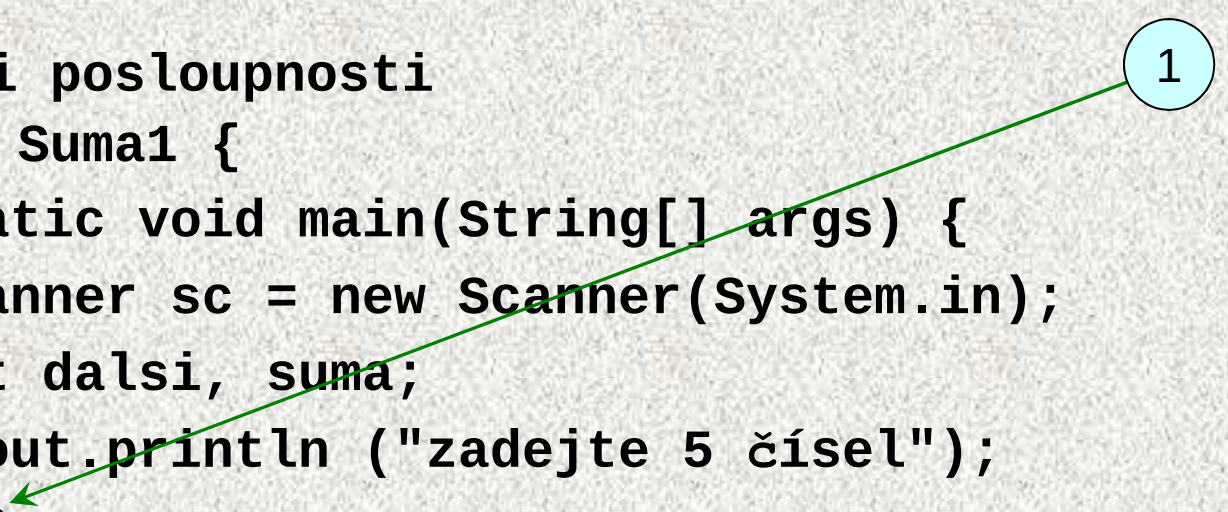



Základní znalost

# JAVA – for /podobné/ (7)

```
import java.util.*;

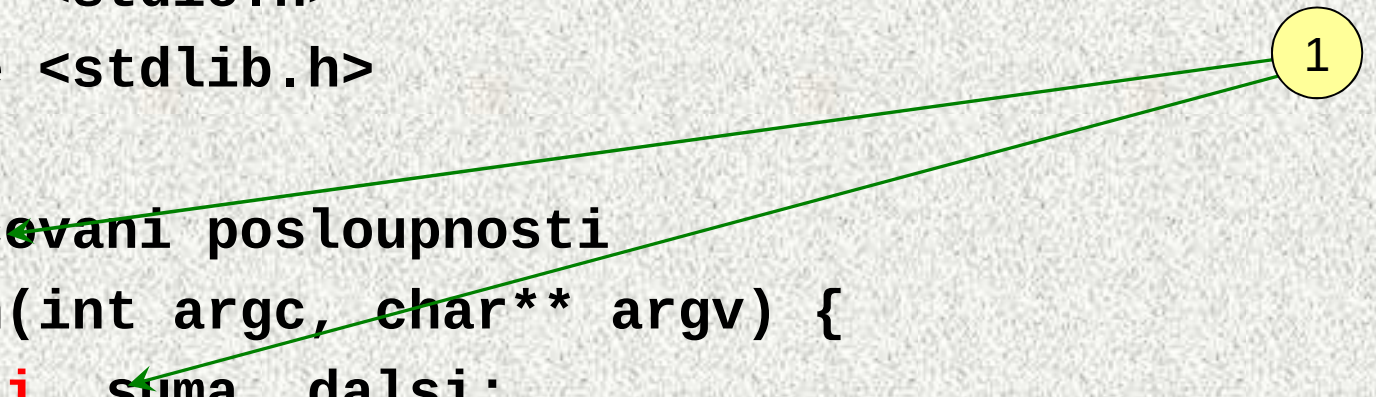
// Zpracovani posloupnosti
public class Suma1 {
   public static void main(String[] args) {
         Scanner sc = new Scanner(System.in);
         int dalsi, suma;
      System.out.println ("zadejte 5 čísel");
      suma = 0;
      for (int i = 1; i <= 5; i++) {
        dalsi = sc.nextInt();
        suma = suma+dalsi;
      }
      System.out.println ("součet je " + suma);
   }
}
```



# C – for /podobné/ (7)

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

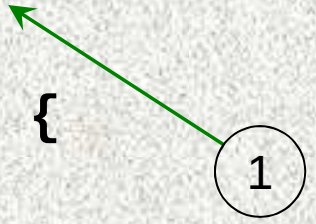
// Zpracovani posloupnosti
int main(int argc, char** argv) {
    int i, suma, dalsi;
    printf("Zadejte 5 cisel \n");
    suma = 0;
    for(i = 1; i <= 5; i++){
        scanf("%d", &dalsi);
        suma = suma + dalsi;
    }
    printf("suma = %d \n\n", suma);
    return (EXIT_SUCCESS);
}
```




Základní znalost

# JAVA – switch, break, continue /shodné/ (8)

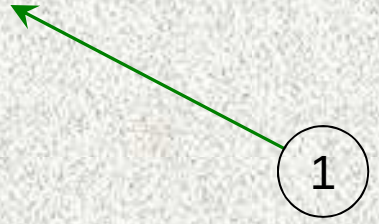
```
public class Prepinac {
   public static void main(String[] args) {
          . . .
     int v, x=6;
     v = 500;
     switch(x) {
       case 2:
           v = 300;
           break;
       case 6:
           v = 400;
           break;
       default:
           v = 0;
     }
   }
}
```




Základní znalost

# C – switch, break, continue /shodné/ (8)

```
// Prepinac
int main(int argc, char** argv){
     int v, x=6;
     . . .
     v = 500;
     switch(x) {
       case 2:
            v = 300;
            break;
        case 6:
            v = 400;
            break;
        default:
            v = 0;
      }
   }
```




Základní znalost

# C a Java - Co C nemá I

***C nemá:***

- Interpret kódu (JVM) (C je kompilovaný)
- Objektovou podporu
    - Třídy, objekty, zapouzdření, dědičnost, polymorfismus
- Jednotnou metodiku vytváření a použití strukturovaných proměnných
    - referenční proměnná, new()
- Automatickou správu paměti
    - Garbage collector
- Velikost proměnných nezávislou na platformě
- Způsob uložení proměnných závislý na OS (Little-endian, Big endian,...)

Základní znalost

# C a Java - Co C nemá II

***C nemá:***

- *Ošetření výjimek metodikou chráněných bloků*
  - **`try, catch, finally`**
- *Standardní podporu grafického uživatelského rozhraní GUI*
- *Standardní podporu řízení událostí (events, listeners)*
- *Standardní podporu webovských aplikací (aplety, síťové připojení)*
- *Standardní podporu (semi)paralelního zpracování úloh*
  - *threads, multitasking, multithreading*

Základní znalost

# C a Java - Co C nemá nebo je jinak I

***v C je jinak:***

- C je kompilovaný jazyk
  - Zdrojový kód je nezávislý (portable) na platformě (málo závislý)
  - Spustitelný kód je závislý na platformě
- Členění programu na moduly, určení jejich vazeb (interface)
- Import knihoven (systémových i uživatelských)
- Určení doby života proměnných, vlastní správa paměti
- Definování konstant a maker

# C a Java - Co C nemá nebo je jinak II

***v C je jinak:***

- Vytváření strukturovaných proměnných (mohou být i statické)
- Určení viditelnosti proměnných, modifikátory přístupu
- Ošetření běhových chyb (run-time errors)
    - odpovědný programátor, překladač nevynucuje
- Správa paměti (heap management)
    - odpovědný programátor, malloc / free
- Práce s booleovskými proměnnými a řetězci (boolean a String není)

# C a Java - Co je v C navíc nebo jinak I

***v C je navíc:***

- Preprocessor
  - Vkládání hlavičkových souborů (header file) do zdrojového kódu
  - Podmíněný překlad
  - Makra
  - #pragma – doplňující příkazy závislé na platformě
- Linker – spojování přeložených modulů a knihoven do spustitelného kódu
- enum – výčtové typy (množina číslovaných pojmenovaných konstant), už je, nebylo
- **Ukazatele (pointer) jako prostředek nepřímého adresování proměnných**

Základní znalost

# C a Java - Co je v C navíc nebo jinak II

***v C je navíc:***

- **`struct`** a bitová pole (strukturované proměnné z různých prvků)
- **`union`** – překrytí proměnných různého typu (sdílení společné paměti)
- **`typedef`** – zavedení nových typů pomocí již známých typů
- **`sizeof`** – určení velikosti proměnné (i strukturovaného typu)
- Jiné názvy i parametry funkcí ze standardních knihoven (práce se soubory, znaky, řetězci, matematické funkce, ….)

# C - Struktura programu

***Program v C se skládá z následujících součástí:***

- Příkazy preprocesoru (preprocessor commands)
- Definice typů (type definitions)
- Prototypy funkcí (function prototypes) kde je uvedena deklarace:
  - Jména funkce
  - Vstupních parametrů
  - Návratové hodnoty funkce
- Proměnné (variables)
- Funkce (functions) (procedura v C je funkce bez návratové hodnoty nebo void)

Základní znalost

# C - Struktura programu

***Funkce v C mají tento tvar:***

- Každý program *musí obsahovat* právě jednu funkci main()
- Všechny funkce (včetně **`main()`**) mají jednotný formát:

```
typ_navrat_hodnoty jmeno_fce (parametry_fce)
{
  lokalni promenne
  prikazy
}
```

- *Uvnitř funkce nelze definovat* lokální funkce (definice funkci nesmí být vnořené)
- Na formátování zdrojového textu překladači nezáleží. Formátování zlepšuje autorovi (i ostatním) čitelnost a potlačuje chyby

Základní znalost

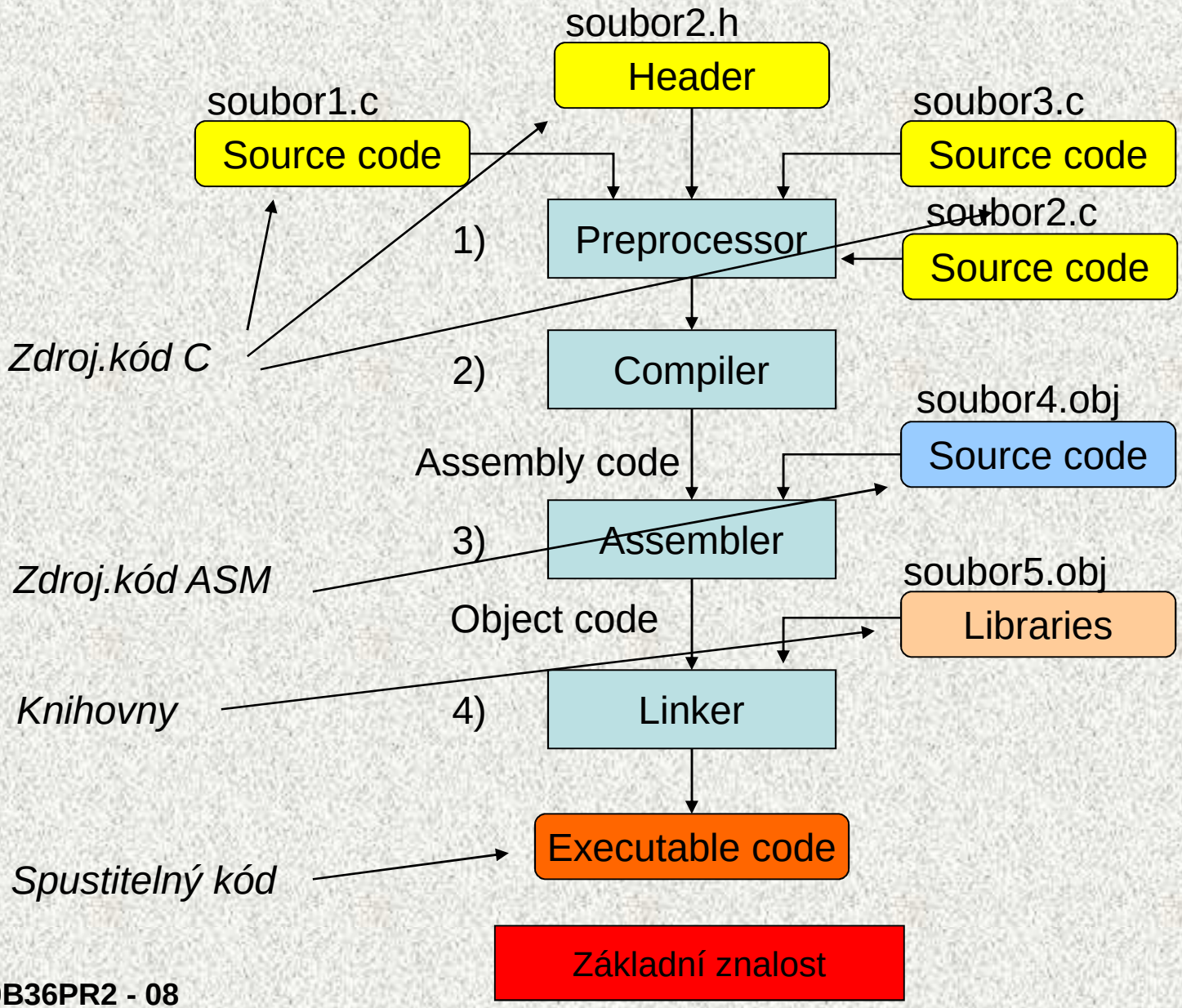
# C - Struktura programu

*Příklad programu v C:*

```
#include <stdio.h> /* hlavičkový soubor */
#include <stdlib.h>
#define NASOBITEL 5 /* symbolicka konstanta */
double funkceNasobeni(double coNasobit);
int main(int argc, char *argv[]) {
   double vysledek; /* lokalni promenna */
   const double PI = 3.14; /* lokalni konstanta */
   printf("Nasobeni\n"); /* vystup na obrazovku
  vysledek=funkceNasobeni (PI);/*volani funkce s parametry */
   printf(" % d * % d = % d\n", NASOBITEL, PI, vysledek);
    /* formatovany vystup */
   return (EXIT_SUCCESS);
}
double funkceNasobeni(double coNasobit) /* definice fce */ {
    double d;
    d = NASOBITEL*coNasobit;
    return ( d); /* vystup hodnoty z funkce */
}
```

Základní znalost

# C - Model kompilace



# C - Model kompilace

***Vytvoření spustitelného kódu (viz obr):***

- 1) Preprocesor:
  - Čte zdrojový kód v C
  - Odstraní komentáře
  - Upraví zdrojový text podle direktiv preprocesoru (řádky začínající #)
    - Vloží do textu obsah jiného souboru #include ….
    - Odebere text vymezený direktivami podmíněného překladu
    - Expanduje makra

- 2) Překladač C:
    - Čte výstup z preprocesoru
    - Kontroluje syntaktickou správnost textu
    - Hlásí chyby a varování
    - Generuje text v assembleru (když nejsou chyby)

Základní znalost

# C - Model kompilace

***Vytvoření spustitelného kódu (viz obr):***

- 3) Assembler:
  - Čte výstup z překladače C
  - Generuje relokovatelný object kód (kód s nevyřešenými odkazy mezi moduly)
  - Přeloží případné moduly zapsané přímo v assembleru (mix programovacích jazyků)

- 4) Linker (spojovací program):
  - Čte object kód všech zúčastněných modulů programu
  - Připojí knihovní object moduly (přeložené dříve nebo dodané),
  - Vyřeší odkazy mezi moduly
  - Generuje spustitelný kód (zjednodušené)

Základní znalost

# Zpracování příkazové řádky

```
int main(int argc, char *argv[])
{
  int i;
  printf("Prikazova radka ma %d retezcu\n", argc);
  for (i = 0;  i < argc;  i++)
    printf("%s\n", argv[i]);
   // scanf("%d", &i);
  return 0;
}
```

Doporučená znalost

# Obsah

Přehled

- C - Struktura programu
- C - Model kompilace
- C - Komentář
- C - Soubor platných znaků (ASCII)
- C – Identifikátory
- C - Datové typy
- C - Konstanty

# C - Komentář

***C používá dva způsoby označení komentáře:***

/* Toto je komentář */

- /* Toto je rovněž komentář
  rozdělený na několik řádků
  */

- // Toto je též komentář až do konce řádku – tento komentář nemusí umět všechny překladače C, v C++ ano

- /* Komentář nesmí být *vnořený* do jiného komentáře
  - **/* to je chybný zápis komentáře */**
  - ***/***

- Preprocesor nahradí každý komentář jednou mezerou
- // ….. tento způsob některé překladače nepodporují

# C - Soubor platných znaků (ASCII)

***C - základní soubor znaků:***

- A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
- a b c d e f g h i j k l m h o p q r s t u v x y z
- 0 1 2 3 4 5 6 7 8 9
- ! “ # % & ‘ ( ) * + , - . / : ; < = > ? [ \ ] ^ ? _ { | } ~
- Oddělovače (whitespace characters)
  space, horizontal tab, vertical tab, newline, form feed
- \0 - null char (koncový znak řetězce)

# C - Soubor platných znaků (ASCII)

***C - základní soubor znaků (pokrač.):***

- Řídicí znaky pro řízení výstupních zařízení (escape sequences)

| Escape sequence | Výstup | | Escape sequence | Výstup |
|---|---|---|---|---|
| \a | Beep (pípnutí) | | \’ | Znak ‘ |
| \b | Backspace | | \” | Znak “ |
| \f | Form feed | | \? | Znak ? |
| \n | Newline | | \\ | Znak \ |
| \r | Carriage return | | \o \oo \ooo<br>( př: \741 ) | o=osmičkové číslice (octal dig) |
| \t | Horizontal tab | | \xh…<br>( př: \x1008 ) | h…=šestnáctkové číslice (hex dig.) |
| \v | Vertical | | | |

Přehled

***C - ostatní znaky:***

- Mohou se použít v komentářích, řetězcích a znakových konstantách

# C - Identifikátory

***C - pravidla pro volbu identifikátorů:***

- Obsahuje A,..,Y,a,..,z,0,..,9 a znak _ (underscore)
- První znak není číslice
- Rozlišují se velká a malá písmena (case sensitive)
- Délka identifikátoru není omezena, ale pouze 31 prvních znaků je významných

  (může se lišit podle implementace)

***C - rezervovaná slova (keywords):***

- Musí být zapsaná malými písmeny

| | | | | | |
|---|---|---|---|---|---|
| auto | default | float | register | struct | volatile |
| break | do | for | return | switch | while |
| case | double | goto | short | typedef | |
| char | else | if | signed | union | |
| const | enum | int | sizeof | unsigned | |
| continue | extern | long | static | void | |

Přehled

# C – Celočíselné datové typy

| Typ | Velikost [byte] | Rozsah | Použití |
|---|---|---|---|
| char | 1 | **-128 až +127 nebo 0 až 255** | Znaky |
| unsigned char | 1 | **0 až 255** | Malá čísla |
| signed char | 1 | **-128 až +127** | Malá čísla |
| int | 2 nebo 4 | **-32.768 až +32.767 nebo**<br>**-2.147.483.648 až +2.147.483.647** | Celá čísla |
| unsigned int | 2 nebo 4 | **0 až 65.535 nebo**<br>**0 až 4.294.967.295** | Kladná celá čísla |
| short | 2 | **-32.768 až +32.767** | Celá čísla |
| unsigned short | 2 | **0 až 65.535** | Kladná celá čísla |
| long | 4 | **-2.147.483.648 až +2.147.483.647** | Velká celá čísla |
| unsigned long | 4 | **0 až 4.294.967.295** | Kladná celá čísla |

- **Typ boolean není (až ANSI C99), =0 ->false, !=0 -> true**

# C – Celočíselné datové typy 2

- Rozsahy celočíselných typů v C nejsou dány normou, ale implementací (pro 16ti a 64 bitové prostředí jsou jiné než je uvedeno v předchozí tabulce
- limits.h, float.h
- Norma pouze garantuje

  *short <= int <= long*

  *unsigned short <= unsigned <= unsigned long*
- Celočíselné literály (zápisy čísel):

| | | | |
|---|---|---|---|
| •dekadický | 123 | 456789 | |
| •hexadecimální | 0x12 | 0xFFFF | (začíná 0x nebo 0X) |
| •oktalový | 0123 | 0567 | (začíná 0) |
| •unsigned | 123456U | | (přípona U nebo u) |
| •long | 123456L | | (přípona L nebo l) |
| •unsigned long | 123456UL | | (přípona UL nebu ul) |

Není-li uvedena přípona, jde o literál typu *int*

# C - Datové typy

***C - racionální čísla (neceločíselné datové typy):***

- Velikost reálných čísel určená implementací
- Většina překladačů se řídí standardem IEEE-754-1985, potom jsou rozsahy reálných čísel dány následující tabulkou:

| Typ | Velikost [byte] | Rozsah (uveden pro kladná č.) | Přesnost |
|---|---|---|---|
| float | 4 | **1.2E-38 až 3.4E+38** | 6 desítkových číslic |
| double | 8 | **2.3E-308 až 1.7E+308** | 15 desítkových číslic |
| long double | 10 | **3.4E-4932 až 1.1E+4932** | 19 desítkových číslic |

***C - typ* `void`*:***

- void značí prázdnou hodnotu nebo proměnnou bez typu (jen ukazatelé)
  - *void* funkce1 (….) - fukce bez návratové hodnoty (procedura)
  - int funkce2 (*void*) - funkce bez vstupních parametrů
  - void *ptr; - ukazatel bez určeného typu (viz dále)

# C - literály

***C - 6 typů konstant:***

- Celočíselné (integer constants)
- Racionální (floating constants)
- Znakové (character constants)
- Řetězcové (literal constants) (literály)
- Výčtové (enumeration constants) (pojmenované prvky množiny)
- Symbolické (#define MAX 300)

# C – Literály racionálních čísel

***C - Formát racionálních literálů:***

- S řádovou tečkou př: 41.9
- Mantisa a exponent př: 5.67E-3 // $5.67*10^{-3}$

  nebo rovněž 5.67e-3

***C - Typ racionálního literálu:***

- Není-li určen explicitně, pak je - *double*
- Určí se explicitně znaky
  - *F nebo f* - *float*
  - *L nebo l* - *long double*
- V tabulce jsou všechny literály typu double

| | | | |
|---|---|---|---|
| **5.19** | **0.519E1** | **0.0519e+2** | **519E-2** |
| **12.** | **12.0** | **.12E2** | **12e0** |
| **370000.0** | **37e+4** | **3.7E+5** | **0.37e6** |
| **0.000004** | **4E-6** | **0.4e-5** | **.4E-5** |

# C - literály- znakové

***C - Formát:***

- Jeden nebo více znaků v *jednoduchých apostrofech*
    - př: ‘B’, ‘g’, ‘ab’, ’9’

***C - Hodnota:***

- Jednoznakový literál má hodnotu odpovídajícího kódu znaku
    - př: v ASCII kódu ‘0’ ~= 48, ‘A’ ~=65
- Mimo ASCII(více než 127) - hodnota závisí na implementaci překladače C

***C - Typ znakové konstanty:***

- Znaková konstanta je typu *int*

***Pozn:***

- Řídicí znaky (escape sequences) mohou tvořit znakové literály
    - př: ‘\n’

# C - řetězcové literály

***C - Formát:***

- Posloupnost znaků a řídicích znaků (escape sequences) *uzavřená v uvozovkách*
    - př: “Já jsem řetězcová konstanta ! \n”

***Pozn:***

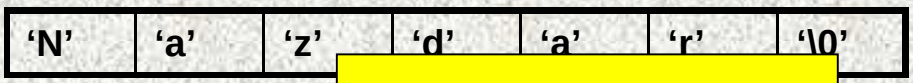
- Řetězcové konstanty oddělené pouze oddělovači (whitespace characters) jsou sloučeny do konstanty jediné.
    - př: “Já jsem” “řetězcová konstanta ! \n”

      se sloučí do

      “Já jsem řetězcová konstanta ! \n”

***C - Typ:***

- Řetězcová konstanta je uložená v *poli typu char* (array of char) a zakončená znakem ‘\0’
    - př: “Nazdar” se uloží v poli typu char takto:



# C - Konstanty - výčtové (enum type)

***C - Formát výčtové konstanty :***

- enum { CERVENA, MODRA, BILA, CERNA};

  pak CERVENA=0 a každý další prvek má hodnotu o jedničku vyšší

- enum { MALY=5, VETSI, VELKY, NEJVETSI }

  pak MALY=5, VETSI=6 a každý další prvek má hodnotu o jedničku vyšší

***C - Typ výčtové konstanty:***

- Výčtová konstanta je typu *int*

# C - #define

***C - Formát:***

- Konstanta se založí příkazem preprocessoru *#define*

  (je to makro bez parametrů, každé #define musí být na samostatné řádku)

  př:

  #define CERVENA 0 /* Zde muze byt komentar */

  **Preprocesor provede textovou náhradu všech výskytů slova CERVENA znakem 0**

  #define MODRA 1 / /Je zvykem jména konstant psát velkými písmeny
- Hodnotu konstanty je možné vyjádřit konstantním výrazem

  př: #define MAX_1 (100*5)-12

- Symbolické konstanty mohou být vnořené

  př: #define MAX_2 MAX_1+30