Úvod do předmětu

Literatura

- Záznamy přednášek a vaše poznámky
- Harbison, S. P., Steele, G. L.: Referenční příručka jazyka C
- Herout, P.: Učebnice jazyka C
- Kernighan, B. W., Ritchie, D. M.: The C Programming Language

Předpokládané znalosti

- nutné:
  - algoritmus, proměnná, přiřazení výraz
  - řídící struktury
  - metody – parametry, návratová hodnoty

- výhodné:
  - metody – volání hodnotou a odkazem
  - pole jako referenční proměnná

Cíle

- číst a psát programy v C s využitím standardních knihoven
- porozumět pointerům
- naučit se využít výhod jazyka C a nepodlehnout jeho nevýhodám

Poznámky

- Když něčemu nerozumím, okamžitě se přihlásím

Historie

- předchůdce = jazyk B
- 1972 – Dennis M. Ritchie, AT&T Bell Labs – jazyk C (odstranění problémů jazyka B)
- masový úspěch – UNIX přepsán celý do C
- 1970 – 1978: K&R Standard („Pre-ANSI C“)
  - shrnuto v [Kernighan, Ritchie]
- 1982 počátek standardizace jazyka a knihoven
- 1989 – American National Standard X3.159-1989 = „ANSI C“ (úprava jazyka dle praktických zkušeností)
- 1990 – převzata pod ISO/IEC 9899-1990
- nejnovější standard ISO/IEC 9899:1999 = „C99)

+ / – jazyka C

+

- multiplatformní – „pro vše, co má CPU, existuje C kompilátor“
- přenositelný – kód pro jednu platformu platný i pro jiné
  - napříč OS, napříč CPU
- nízkoúrovňový – přímý přístup do paměti, bitové operace
- rychlý, vhodný pro systémové programování (vývoj OS, Linux napsán kompletně v C)
- standardizovaný

–

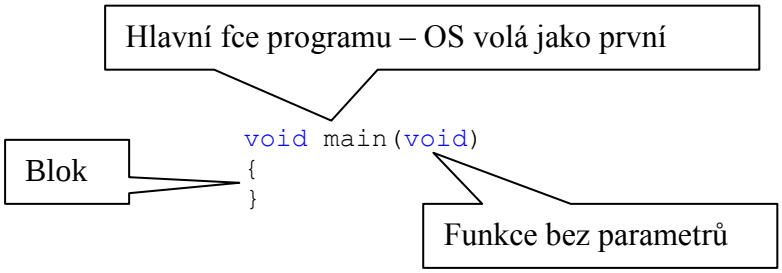
- nebezpečný – slabá typová kontrola, nejednoznačná syntaxe
- nízkoúrovňový

Kompilátory

- pro PC (Intel x86)
- Intel C compiler
- GCC (GNU Compiler Collection)
  - kvalitní, free (GPL licence) nejen C/C++ kompilátor
  - napříč OS (Windows, Linux, Unix, Solaris, ...)
  - napříč CPU (x86, ARM, AVR, SPARC, ...)
  - 100% C99 kompatibilní
- Microsoft C/C++ compiler
  - C99 pouze částečně, ANSI C jen tak tak ☺
- Kompilátor vs. IDE
  - IDE pouze frontend kompilátoru

První programy

- Nejjednodušší program



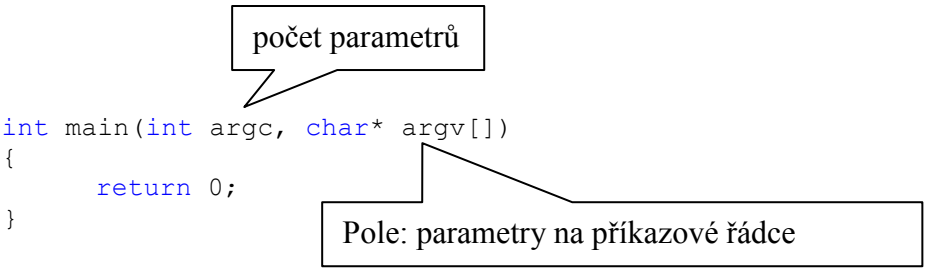
- program s předáním informací o svém běhu OS

```
int main(void)
{
	return 0;
}
```

Předána OS – lze testovat v dávkových souborech (*.bat)

  - v MS-DOS proměnná `ERRORLEVEL`, 0 = OK

- Program s parametry na příkazovém řádku

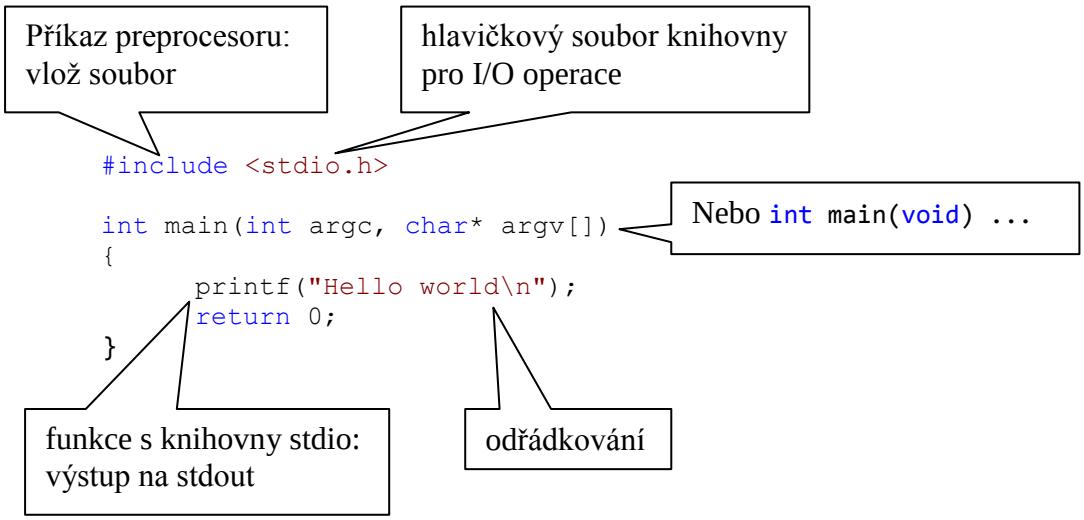


  - Příklad příkazové řádky:

```
C:\adresar\program.exe param1 param2
```

```
argc: 3
argv[0]: "C:\adresar\program.exe"
argv[1]: "param1"
argv[2]: "param2"
```

- „Hello Word“



  - hlavičkový soubor (header file), „hlavička“
  - stdio = STandarD Input Output
  - stdout = STandarD OUTput = standardní výstup, obvykle obrazovka („konzole“, příkazový řádek)

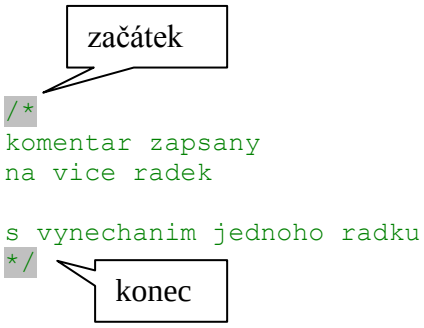
Zdrojový kód

- txt soubor
- (obvykle *.c)
- bílé znaky: ENTER, mezera, tabelátor ... → překladač ignoruje

Základní syntaxe

- dle ANSI C
- case sensitive
- příkazy ukončeny ;
- všechna klíčová slova a funkce ze standardní knihovny = malá písmena
- řetězce `"v uvozovkách"`
- znaky v apostrofech `'x'`

Komentáře



```
/* komentar na jednom radku */

vysledek = a /* promenna */ + b;
```

  - nelze vhnízděné komentáře
    ```
    /* aaaa /* bbb */ aaa */
    ```

- C99 (C++) komentáře – navíc `//`
  - konec = konec řádku

```
vysledek = a + b;       // nejaka operace
```

Klíčová slova

- 32 slov

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

Identifikátory

- case sensitive (`Plat`, `PLAT`, `plat` = různé)
- význam pouze prvních 31 znaků (zbytek překladačem ignorován)
- lze použít:

  aA – zZ

  _

  0 až 9 (nesmí být na začátku)

Základní datové typy

Celočíselné

lze pouze `short`

lze pouze `long`

	Win32 (Linux)	MS-DOS (Turbo C)	AVR (GCC)
`char`	8 b	8 b	8 b
`short int`	16 b	16 b	16 b
`int`	32 b	16 b	16 b
`long int`	32 b	32 b	32 b

- vše buď
  - `unsigned` – bez znaménka
  - `signed` – se znaménkem (implicitně, kromě `char` – dle překladače)

- shrnutí – příklady ekvivalentních zápisů:

```
short       =     short int    =     signed short int
int         =     signed int
long        =     long int     =     signed long int
```

- poznámky
  - C99 – `long long` = 64 b
  - velikost datových typů – vždy platí:

    `char` = 8 b
    `unsigned` = `signed`
    `short` <= `int` <= `long`

Reálné

`float`	32 b	dle IEEE 754, jednoduchá přesnost
`double`	64 b	dle IEEE 754, dvojnásobná přesnost
`long double`	64 b	Win 32 = `double`

Zjištění velikosti

- operátor `sizeof(Typ)`
- vrací velikost typu v B

limits.h

- parametry celočíselných datových typů

```
// ...
#define CHAR_BIT      8         /* number of bits in a char */
#define SCHAR_MIN   (-128)      /* minimum signed char value */
#define SCHAR_MAX     127       /* maximum signed char value */
#define UCHAR_MAX     0xff      /* maximum unsigned char value */
// ...
```

float.h

- parametry reálných datových typů

```
// ...
#define DBL_DIG         15    /* # of decimal digits of precision */
#define DBL_EPSILON     2.2204460492503131e-016
      /* smallest such that 1.0+DBL_EPSILON != 1.0 */
#define DBL_MANT_DIG    53    /* # of bits in mantissa */
#define DBL_MAX         1.7976931348623158e+308 /* max value */
#define DBL_MAX_10_EXP  308   /* max decimal exponent */
#define DBL_MAX_EXP     1024  /* max binary exponent */
#define DBL_MIN         2.2250738585072014e-308
      /* min positive value */
// ...
```

Zajímavost

- The International Obfuscated C Code Contest
- vítězové http://fly.srk.fer.hr/ioccc/winners.html
- 1998 – David Lowe

```
#include <stdio.h>
#define PO(o,t)\
(((o>64)&&(o<91))?(((t>96)&&(t<123))?(t-32):(t)):(((t>64)&&(t<91))?(t+32):(t)))

   void main( ) {                                          char *poo="poot",
   *Poo="pootpoot"       ,O[9];int      o,t,T,p;(t=p     =0)||(*O='\0');while
   ((o=       getc(    stdin    ))!=   (EOF))if ((p==     0 ) && (((o>64)&&(
   o<91       )) ||    (( o>    96 )   &&(o<     123)          ))) (
   t!=8       )&&(O    [t]=     o)&&   (O[++     t] =          '\0')
   ;else {if (t>7)     {for     (T =   0 ; T     <=7;          T++ )
   printf("%c",        PO(*(    O+T),  *(Poo+    T)));         printf
   ("%c",               o);}else if     (t>3){for (T          =0;T<=
   3;T++)                                                    printf
   ("%c",                                                    PO(*(O
   +T),*(                                                    poo+T)

) ) ;  printf( "%c" , o ) ; } else  printf ( "%s%c" , O , o ) ; ( t =  0 ) || (
* O =  '\0' ) ; ( o == 60 ) && ( ++p ) ; ( o == 62 ) && (p!=0) && ( --p ) ; } }
```

Rozšíření jazyka

- nad rámec standardů
- každý kompilátor své vlastní, viz dokumentace
- Př.: Microsoft specific – výjimky v C

```
puts("hello");
__try
{
    puts("in try");
    __try
    {
        puts("in try");
        RAISE_AN_EXCEPTION();
    }
    __finally
    {
        puts("in finally");
    }
}
__except( puts("in filter"), EXCEPTION_EXECUTE_HANDLER )
{
    puts("in except");
}
puts("world")
```