Mikroradiče

Jazyk C pre mikroradiče - úvod

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY
ÚSTAV ELEKTRONIKY A FOTONIKY
Laboratórium DSP a mikroradičov

:: Jazyk C

- 1972 – objav C (Dennis Ritchie – AT&T Bell Labs)
- 1978 – publikovaná kniha „The C Programming Language“ – prvá špecifikácia jazyka
- 1989 – C89 štandard (známy ako ANSI C, resp. Standard C)
- 1990 – ANSI C prebraný ISO, známy ako C90
- 1999 – C99
- 2007 – práca na novom štandarde C - C1X

:: Jazyk C

- C je minimalistický programovací jazyk na úrovni blízkej hardvéru
- Je určený pre systémové programovanie:
  - *OS (Linux), mikroradiče, DSP procesory: digitálne audio a TV systémy*

- Výhody
  - jednoduché jadro s podporou knižníc a minimalistickou množinou kľúčových slov
  - štruktúrované programovanie s jednoduchým systémom dátových typov
  - nízkoúrovňový prístup k pamäti pomocou smerníkov

- Nevýhody
  - nedostatočná typová kontrola
  - chýbajúca kontrola rozsahov premenných
  - automatické čistenie pamäte

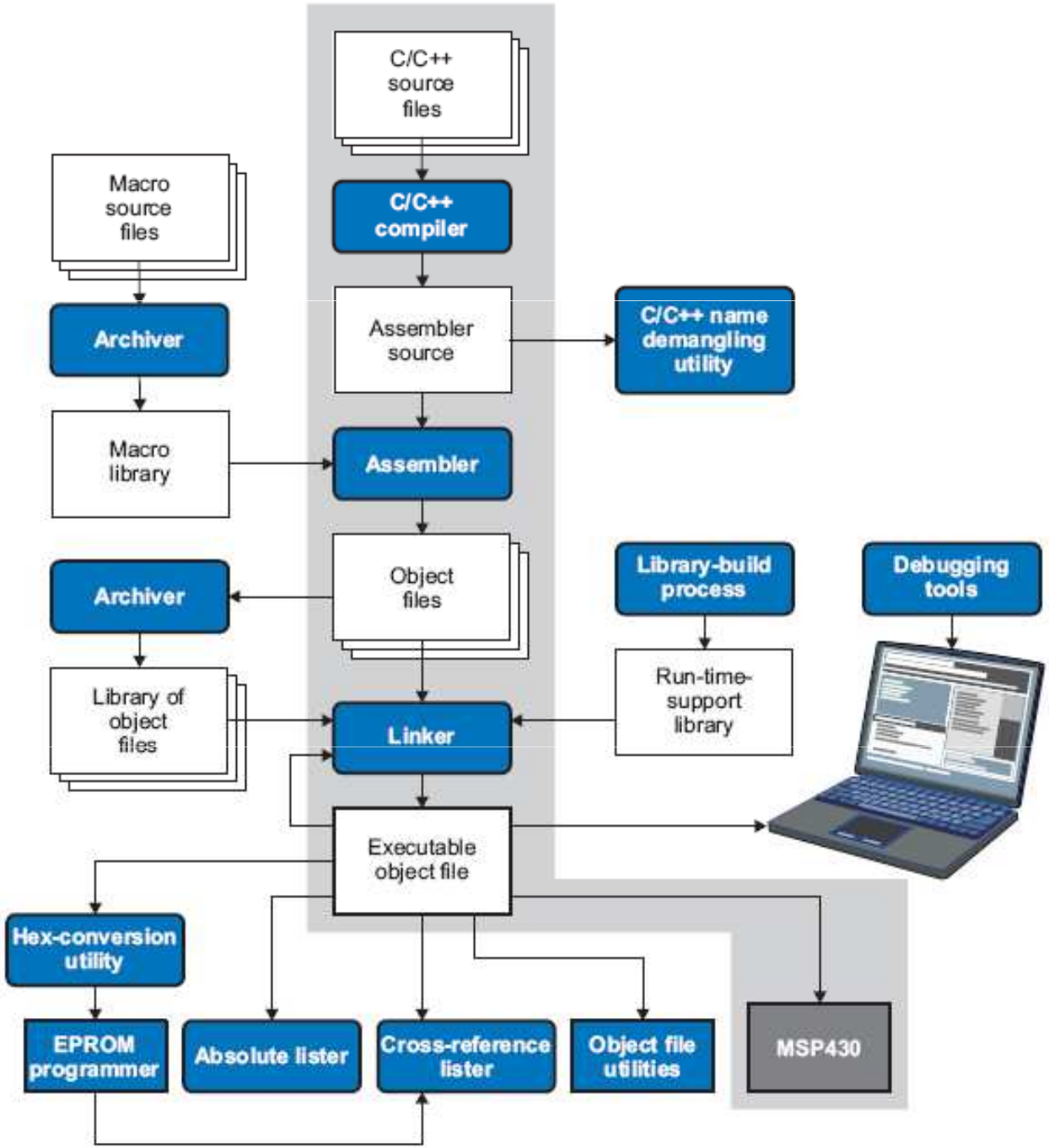
C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it blows your whole leg off.

Bjarne Stroustrup

computer scientist and creator of the C++ programming language

- Programy pre malé systémy založené na mikroradičoch obvykle neobsahujú komplikované manipulácie so zložitými dátovými objektami.
- Zdrojový kód je skôr venovaný riadeniu periférnych modulov mikroradiča prostredníctvom riadiacich registrov a obsluhe interných a externých udalostí (prerušení).
- Dôležité sú bitové, bytové a slovné operácie.

:: Postupnosť pri vývoji softvéru pre procesory



:: Štruktúra .c súborov

/ Začíname komentárom o obsahu súboru */*

Príkazom ***#include*** *vložíme požadované knižnice*

Uvedieme ***prototypy funkcií*** *a* ***deklarácie premenných***

definícia funkcie main()
{
    telo funkcie
}

definícia ďalšej funkcie
{
    telo funkcie
}
...

:: Komentáre

- komentáre: */∗ toto je jednoduchy komentar ∗/*
- komentár môže zaberať viacero riadkov:

  / ∗ tento komentar
  zabera
  niekolko riadkov∗ /

- kompilátor komentáre ignoruje
- môžu sa objaviť kdekoľvek v kóde

:: Hlavičkové súbory

- hlavičkové súbory: konštanty, funkcie, deklarácie

  #include <stdio.h> – prečíta obsah súboru stdio.h
  #include <msp430f169.h> – konštanty, názvy registrov...

- stdio.h: štandardné I/O funkcie

- ďalšie dôležité hlavičkové súbory: math.h, string.h...

:: Deklarácia a inicializácia premenných

- všetky premenné musíme pred použitím v kóde deklarovať:

```
int n;
float pi;
```

int – celočíselný dátový typ
float – dátový typ s plávajúcou desatinnou čiarkou

- neinicializovaná premenná môže obsahovať ľubovoľnú hodnotu

- inicializovať môžeme aj počas deklarácie:

```
float pi = 3.1415927;
```

- deklarovať a inicializovať môžeme viacero premenných naraz:

```
int a, b, c =0, d=4;
```

:: Výrazy

- predpokladajme, že x a y sú premené
- príklady výrazov:

  x+y, x-y, x*y, x/y, x%y

- jednoduchý príkaz:

  y = x+3∗x/(y−4);

- vo výrazoch môžeme používať numerické hodnoty
- bodkočiarka ukončuje príkaz (nie riadok)
- x += y, x -= y, x *= y, x /= y, x %= y

:: Poradie operácií

- poradie operácií:

operátor	poradie vykonávania
+, - (znamienko)	R -> L
*, /, %	L -> R
+, -	L -> R
=,+=,-=,*=,/=,%=	R -> L

- zátvorkami môžeme poradie operácií upraviť
- pri pochybnostiach o priorite použijeme vždy zátvorky

:: Poradie operácií

- predpokladajme, že x =2.0 a y =6.0
- vyriešte príkaz float z = x + 3 ∗ x / (y − 4);

:: Poradie operácií

- predpokladajme, že x =2.0 a y =6.0
- vyriešte príkaz float z = x + 3 ∗ x / (y − 4);
- 1. najskôr vyjadríme výraz v zátvorkách
  float z = x+3∗x/(y−4); float z = x+3∗x/2.0;
- 2. určíme násobky a podiely (L -> R)
  float z = x+3∗x/2.0; float z = x+6.0/2.0; float z = x+3.0;
- 3. vypočítame súčty a rozdiely
  float z = x+3.0; float z = 5.0;

:: Prototypy funkcií

- všetky funkcie v zdrojovom kóde musia byť pred použitím tiež deklarované
- túto deklaráciu nazývame prototyp funkcie

```
int factorial (int);
int factorial (int n);
```

- prototypy mnohých funkcií sa nachádzajú v štandardných hlavičkových súboroch
- prototypy vlastných funkcií je vhodné umiestniť do hlavičkoveho súboru a pomocou **#include** ho pripojiť k súboru obsahujúcemu funkciu **main()**

```
#include <display.h>
```

:: Prototypy funkcií

- všeobecná forma prototypu funkcie:

```
navratovy_typ nazov_funkcie(arg1, arg2, ...);
```

- argumenty: lokálne premenné, hodnoty, ktoré posiela volajúca funkcia

- návratová hodnota: hodnota, ktorá sa vráti volajúcej funkcii, keď volaná funkcia skončí

- void – prázdny typ, ktorým oznamujeme kompilátoru, že funkcia nemá návratovú hodnotu a/alebo argumenty

```
int rand(void);
```

:: Funkcia main()

- main(): vstupný bod programov v jazyku C
- dá sa povedať, že main() je hlavná slučka programu?
- najjednoduchšia verzia:
  - žiadne vstupy (prečo?)
  - výstup 0 pri úspešnom ukončení a nenulová hodnota v prípade chyby (má toto v prípade mikroradičov zmysel?)

```
int main(void);
{
}

void main(void);{}
```

:: Definícia funkcie

definícia funkcie
{
deklarácia lokálnych premenných;
príkazy funkcie;
}

- musí súhlasiť s prototypom (ak existuje)
- názvy premenných nemusia súhlasiť s prototypom
- za definíciou funkcie nie je bodkočiarka
- zložené zátvorky definujú blok – ucelenú oblasť kódu
- premenné deklarované v bloku existujú iba v danom bloku
- deklarácie premenných musia predchádzať príkazom

Dátový typ

Dátový typ objektu v pamäti určuje rozsah hodnôt, ktoré môže daný objekt obsahovať a typ operácií, ktoré je možné s objektom použiť

Operátory

Operátor špecifikuje manipuláciu s objektom
(napr. numerické operácie vs. operácie s reťazcami)

Rozoznávame tri druhy operátororov:

- unárne (napr.: --,++)
- binárne (napr.: +,-,*,/)
- ternárne (?:)

Výrazy

Výraz v programovacom jazyku predstavuje kombináciu hodnôt, premenných, operátorov a funkcií.

Premenná

Premenná je názov referencie k hodnote uloženej v pamäti alebo výrazu, ktorého hodnotu je možné určiť.

Napr.:

```
int x = 0, y = 0; y = x + 2;.
```

x, y sú premenné
y = x + 2 je výraz
+ je operátor

:: Pravidlá pre tvorbu názvov premenných

1. Názvy premenných môžu obsahovať

   - písmená,
   - číslice,
   - podčiarkovník (znak _) (break character)

2. Názvy premenných musia začínať písmenom.

3. Kľúčové slová jazyka C nemôžeme použiť ako názvy premenných.

4. Názvy premenných sú citlivé na veľkosť písmen:

   int x; int X deklaruje dve rôzne premenné

:: Tvorba názvov premenných - kvíz

```
int money$owed

int total_count

int score2

int 2nd_score

int while
```

:: Tvorba názvov premenných - kvíz

int money$owed; (nesprávne: nemôže obsahovať $)

int total_count (správne)

int score2 (správne)

int 2nd_score (nesprávne: musí začínať písmenom)

int while (nesprávne: premenná nemôže byť kľúčovým slovom)

:: Dátové typy

Jazyk C má iba malý súbor dátových typov!

1. Numerické dátové typy (short, int, long, float, double)
2. Znakový dátový typ (char)
3. Užívateľsky definované dátové typy (struct, union)

1. Neznamienkové numerické dátové typy majú približne dvojnásobný rozsah oproti znamienkovým.

2. Znamienkové a neznamienkové znaky sa odlišujú iba v prípade, keď sú použité v aritmetických výrazoch.

	signed	unsigned
short	**short int** x;**short** y;	**unsigned short** x;**unsigned short int** y;
default	**int** x;	**unsigned int** x;
long	**long** x;	**unsigned long** x;
float	**float** x;	N/A
double	**double** x;	N/A
char	**char** x; **signed char** x;	**unsigned char** x;

:: MSP430 C/C++ EABI

Type	Size	Alignment	Representation	Range	
				Minimum	Maximum
signed char	8 bits	8	Binary	-128	127
char	8 bits	8	ASCII	0 or -128 (1)	255 or 127 (1)
unsigned char	8 bits	8	Binary	0	255
bool (C99)	8 bits	8	Binary	0 (false)	1 (true)
_Bool (C99)	8 bits	8	Binary	0 (false)	1 (true)
bool (C++)	8 bits	8	Binary	0 (false)	1 (true)
short, signed short	16 bits	16	2s complement	-32 768	32 767
unsigned short	16 bits	16	Binary	0	65 535
int, signed int	16 bits	16	2s complement	-32 768	32 767
unsigned int	16 bits	16	Binary	0	65 535
long, signed long	32 bits	16	2s complement	-2 147 483 648	2 147 483 647
unsigned long	32 bits	16	Binary	0	4 294 967 295
long long, signed long long	64 bits	16	2s complement	-9 223 372 036 854 775 808	9 223 372 036 854 775 807
unsigned long long	64 bits	16	Binary	0	18 446 744 073 709 551 615
enum	varies (2)	16	2s complement	varies	varies
float	32 bits	16	IEEE 32-bit	1.175 494e-38 (3)	3.40 282 346e+38
double	64 bits	16	IEEE 64-bit	2.22 507 385e-308 (3)	1.79 769 313e+308
long double	64 bits	16	IEEE 64-bit	2.22 507 385e-308 (3)	1.79 769 313e+308

:: Užitočné referencie

1. Kernighan B., Ritchie D.: Programovací jazyk C, Computer Press, 2006, ISBN: 80-251-0897-X

2. Herout, P.: Učebnice jazyka C, 4. vydání, Kopp, 2004,ISBN 80-7232-220-6

3. Texas Instruments, Inc.: MSP430 Optimizing C/C++ Compiler v.4.2 User's Guide (Rev. H)

Koniec prednášky

Jazyk C pre mikroradiče - úvod