Linguagem C

Fernando Albuquerque

061-2733589

fernando@cic.unb.br

Fernando Albuquerque - fernando@cic.unb.br

Tópicos

- Linguagem C Padrão
- Identificadores
- Diretivas
- Comentários
- Variáveis e constantes
- Tipos básicos
- Operadores e expressões
- Controle de fluxo
- Controle de iteração
- Matrizes
- Ponteiros
- Estruturas de dados
- Uniões
- Enumerações
- Funções
- Biblioteca padrão

Fernando Albuquerque - fernando@cic.unb.br

Linguagem C Padrão

- Conceitos :
  - » American National Standards Institute (ANSI)
  - » Padrão internacional ISO/IEC 9899
  - » Define a linguagem C de forma precisa
  - » Possibilita maior portabilidade
- Compilando com C padrão :

```
cc -Xc  subs.c  main.c  -o prog
```

chave arquivos fonte programa executável

Identificadores

- Palavras reservadas :

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

Diretivas

- Diretivas :
  - » Controlam ação do pré-processador
  - » São executadas operações sobre o código fonte
  - » Identificadas através do caracter #
  - » Terminadas ao final da linha
  - » Se precisar continuar após final da linha usado \
  - » Pode aparecer em qualquer parte do código fonte
  - » Efeito dura até que nova diretiva cancele operação

Diretivas

- #define
  - » Informa que identificador deve ser substituído
  - » Resulta em código mais fácil de entender e adaptar
  - » Pode ser usada na definição de macros
  - » Macros são expandidas pelo pré-processador
  - » Uso de macros resulta em ganhos de performance
  - » Uso de macros resulta em maior gasto de memória
  - » Parâmetros substituem diretamente os formais
  - » Cancelada a definição via # undef

Diretivas

- # include
  - » Insere conteúdo do arquivo identificado
  - » Nome pode se encontrar entre < > ou “ “
  - » Podem ser aninhadas diversas diretivas # include
  - » Comum o uso em arquivos de cabeçalho
- Exemplo :
  - » # include < stdio.h >
  - » # include “constantes”

Diretivas

- Compilação Condicional:
  - # if expressão
  - # elif expressão
  - # ifdef identificador
  - # ifndef identificador
  - # else
  - # endif

Fernando Albuquerque - fernando@cic.unb.br

Comentários

- Comentários :
  - » Podem conter qualquer mensagem entre /* e */
  - » Não é permitido o aninhamento de comentários
  - » Deve ser adotada uma formatação adequada
- Exemplos :

```
/*
/* Primeira sugestao de formatacao do texto de comentario
*/

/* Segunda sugestao de formatacao  do texto de comentario */
```

Fernando Albuquerque - fernando@cic.unb.br

Tipos Básicos

- Caracteres :
  - » char nome ; - caracter sinalizado (?)
  - » signed char nome; - caracter sinalizado
  - » unsigned char nome ; - caracter não sinalizado
- Inteiros :
  - » int nome;- short sinalizado;
  - » short nome;- short sinalizado;
  - » short int nome;- short sinalizado;

Tipos Básicos

- unsinged int nome; - short não sinalizado
- unsigned short int nome; - short não sinalizado
- long int nome; - long sinalizado
- long nome; - long sinalizado
- unsigned long nome; - long não sinalizado

- Ponto-flutuante :
  - float nome;
  - double nome;
  - long double nome;

Operadores e Expressões

- Expressões e operadores unários :
  - » sizeof nome - tamanho de uma variável
  - » sizeof (nome) - tamanho de um tipo
  - » Incremento (++) e decremento (--)
  - » + e -
  - » Negação a nível de bit (~)
  - » Endereço (&)
  - » Indireção (*)
  - » Força tipo (tipo)

Fernando Albuquerque - fernando@cic.unb.br

Operadores e Expressões

- Aritméticos :
  - Multiplicação *
  - Divisão /
  - Módulo %
  - Adição +
  - Subtração -
- Relacionais :
  - Igual a ==
  - Não igual a !=
  - Menor que <
  - Maior que >
  - Menor ou igual <=
  - Maior ou igual >=

Fernando Albuquerque - fernando@cic.unb.br

Operadores e Expressões

- Lógicos :
  - E &&
  - OU ||
  - Não !
- A nível de bits :
  - E &
  - OU |
  - Não ~
  - XOR ^
  - Shift << e >>

Fernando Albuquerque - fernando@cic.unb.br

Controle de Fluxo

- if - else
  - » Possibilita decidir que enunciado executar
  - » Teste é verdadeiro se valor diferente de zero

```
if ( condição )
{
      enunciados;
}
else
{
      enunciados;
}
```

Enunciados executados se teste for verdadeiro

Enunciados executados se teste for falso

Controle de Fluxo

- switch - case :
  - » Substitui enunciados if-else aninhados
  - » Possibilita seleção de conjunto de enunciados

```
switch ( expressão )
{
        case constante_1 :      enunciado_1;
                                break;
        case constante_2 :      enunciado_2;
                                break;
        default :               enunciado_3;
                                break;
}
```

Controle de Fluxo

- Operador condicional ? :
  - » Substitui a instrução if em certas situações

```
nome = condição ? expressão_1 : expressão_2 ;
```

  - » Equivale aos seguintes enunciados

```
if  ( condição )
        nome = expressão_1;
else
        nome = expressão_2;
```

Fernando Albuquerque - fernando@cic.unb.br

Controle de Iteração

- while :
  - » Possibilita execução repetida de um enunciado

```
while ( condição )
{
        enunciados;
}
```

- do-while :
  - » Repetição ocorre pelo menos uma vez

```
do {
        enunciados;
} while ( condição );
```

Fernando Albuquerque - fernando@cic.unb.br

Controle de Iteração

- for :
  - » Execução enquanto valor dentro de uma faixa

```
for ( expressão_1 ; expressão_2 ; expressão_3 )
enunciado;
```

  - » Eqüivale aos seguintes enunciados :

```
expressão_1;
while ( expressão_2 )
{
        enunciado_1;
        expressão_3;
}
```

Controle de Iteração

- break :
  - » Desvia fluxo para enunciado seguinte a um loop controlado por switch , while , do-while ou for.
- continue :
  - » Finaliza apenas a iteração em execução, desvia o fluxo de volta para expressão de condição.
- return :
  - » Retorna da execução de uma função, quando do retorno pode ser passado, ou não , um valor.

Ponteiros

- Exemplos de aplicações :
  - » Possibilita que , através da passagem do endereço de uma variável , a mesma possa ter o seu valor acessado e modificado pelo código presente em uma função
  - » Possibilita versatilidade no acesso a estruturas de dados tais como listas e árvores
- Declaração :

```
int      dado_a , dado_b  , *pont_a , *pont_b;
```

Fernando Albuquerque - fernando@cic.unb.br

Estruturas de Dados

- Conceitos :
  - » Agrupamento de variáveis relacionadas
  - » Facilita organização dos dados
- Definição :

```
struct  nome_da_estrutura
{
        tipo_membro_1   nome_membro_1 ;
        tipo_membro_2   nome_membro_2;
};
```

Fernando Albuquerque - fernando@cic.unb.br

Uniões

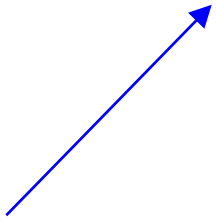
- Conceitos :
  - » Membros são mutuamente exclusivos
  - » Compartilham um mesmo espaço de memória
  - » Total de memória igual ao maior membro
- Definição :

```
union  nome_da_union
{
        tipo_membro_1   nome_membro_1 ;
        tipo_membro_2   nome_membro_2;
};
```

Enumerações

- Conceitos :
  - » Possível definir tipos que pode assumir apenas valores pré-especificados
- Definição :

enum nome_da_enumeração { lista_de_valores } ;



Relação de valores que podem ser atribuídos à enumeração

Funções

- Definição :

```
classe tipo_retorno nome_da_função ( tipo par_1 , ... )
{
        corpo da função
        return ( dado );
}
```

- Protótipo :

```
tipo_retorno  nome_da_função ( tipo par_1 , ... );
```

Fernando Albuquerque - fernando@cic.unb.br

Biblioteca Padrão

- Exemplos de funções da biblioteca padrão :

abort	abs	acos	asctime	asin	atan	atan2
atexit	atof	atoi	atol	bsearch	ceil	calloc
clearerr	clock	cos	cosh	ctime	difftime	div
exit	exp	fabs	fclose	feof	ferror	fflush
fgetc	fgetpos	fgets	floor	fmod	fopen	fprintf
fputc	fputs	fread	free	freopen	frexp	fscanf
fsetpos	ftell	fwrite	getc	getchar	getenv	gets
gmtime	isalnum	isalpha	iscntrl	isdigit	isgraph	islower
isprint	ispunct	isspace	isupper	isxdigit	labs	ldexp
ldiv	localeconv	localtime	log	log10	longjmp	malloc
mblen	mbstowcs	mbtowc	memchr	memcmp	memcpy	memmove