Introdução a Linguagem C

Prof. Msc. Carlos Alberto

Agenda

- Introdução
- Primeiros Passos
- “HelloWorld”
- Introdução às Funções

Introdução

- Linguagem de programação baixo nível ou alto nível?

Introdução

- Linguagem de Programação
  - Características de “Alto nível”
  - Características de “Baixo nível”
  - Versátil
  - Compilada

Introdução

- Vantagem
  - A linguagem C permite criar programas extremamente rápidos e eficientes sem ter que recorrer ao Assembly.
- Desvantagem
  - A verificação de erros que o programador deve fazer é grande. (Liberdade na programação)

Primeiros Passos

- C é "Case Sensitive"
  - Diferencia letras minúsculas e MAIÚSCULAS
  - EX.:
    soma ≠ Soma ≠ SOMA ≠ SoMa

Linguagem Estruturada

- C é uma linguagem estruturada
  - Compartimentalização do código e seus dados
    - Seccionar o código
    - Sub-rotinas
      - Variáveis Locais Temporárias
    - Programador precisa saber apenas o que uma função faz e não como ela faz
    - O uso do **goto** é proibido ou fortemente desencorajado
    - **O principal componente estrutural de C é a função**

Funções

- Um programa em C é uma coleção de uma ou mais funções.

- Forma Geral de uma função

```
tipo_de_retorno nome_da_função (lista de parametros){
        Corpo da função;
}
```

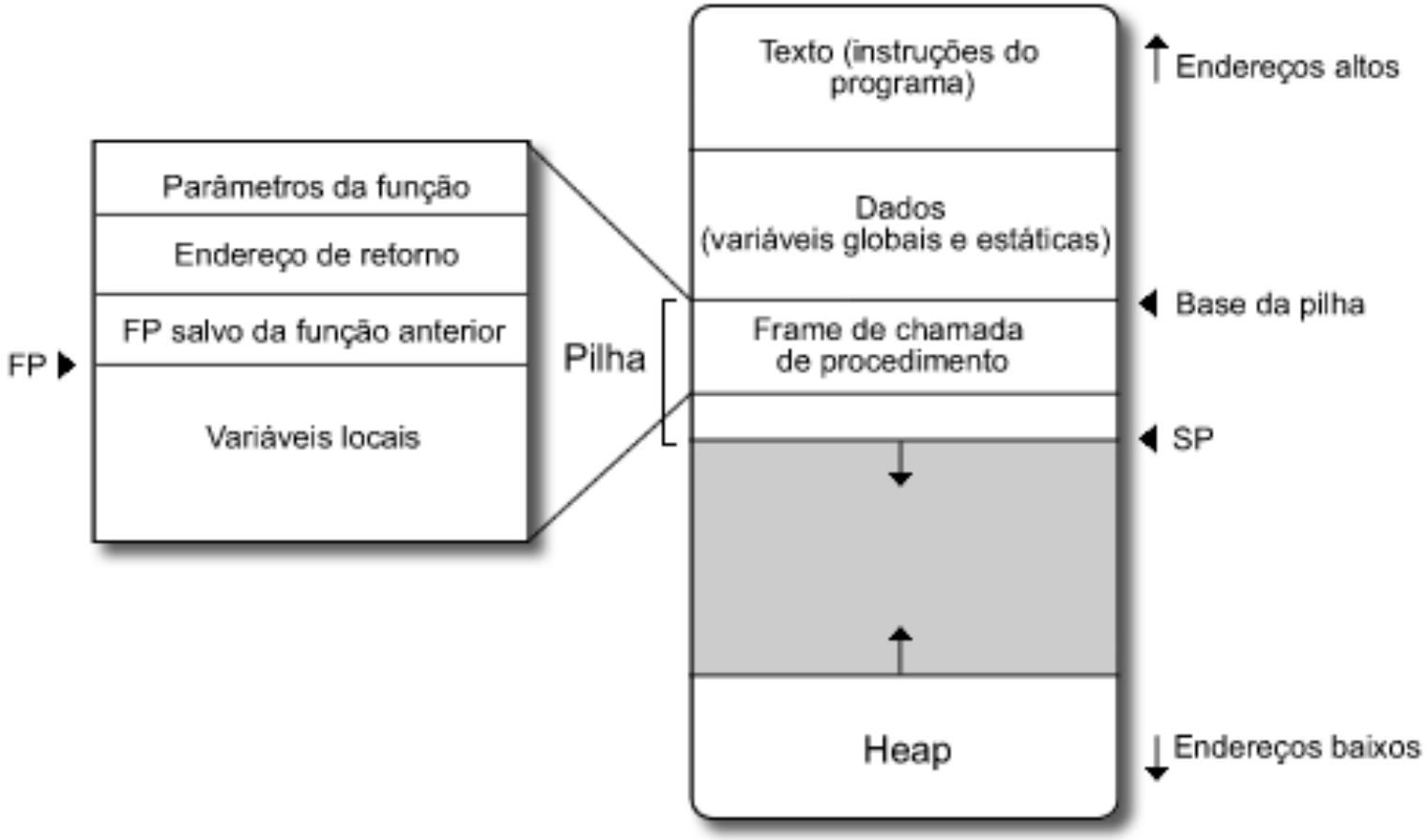
Exemplo:

```
void multiplica (int a , int b){
    printf("%d", a *  b);
}
```

Mapa de Memória

- Um programa C compilado cria quatro regiões de memória
  - Código do Programa
  - Variáveis Globais
  - Heap
    - Região de memória livre utilizável pelo programa
      - Ex. Alocação dinâmica
  - Pilha
    - Endereços de retorno das chamadas de função
    - Estado atual da CPU
    - Argumentos para funções
    - Variáveis Locais

Mapa de Memória



Fonte: http://www.cic.unb.br/~rezende/trabs/buffer_overflow_files/processo.png , acessado em 08/08/2013

Variáveis

- Posição de memória “nomeada”
- Devem ser inicializadas antes de serem usadas
- Locais
- Globais

HelloWorld

- Estrutura de um programa em C

```
#include <stdio.h>
/* Um Primeiro Programa */
int main ()
{
    printf (“Hello World!\n");
    return(0);
}
```

Palavras Reservadas em C

- Atenção C é CASE-SENSITIVE

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

Tipos de Dados

- Tipos de Dados
  - Variáveis e Constantes
    - Os primeiros 32 caracteres de um nome identificador são significantes.
    - Identificadores são usados para referenciar variáveis, funções , rótulos , etc.

Tipos de Dados

- Tipos Básicos
  - Char
  - Int
  - Float
  - Double
  - Void

String

- A linguagem C não possui o tipo String.
  - Strings são armazenadas em cadeias de caracteres
  - Exemplo: char nome[31];

Modificadores de Tipo

- Modificadores de Tipos
  - signed
  - unsigned
  - long
  - Short

Observação: unsigned x;
unsigned int x; //As expressões são equivalentes

Modificadores de Tipo

Tipo	Tamanho aproximado em bits	Faixa mínima
char	8	-127 a 127
unsigned char	8	0 a 255
signed char	8	-127 a 127
int	16	-32.767 a 32.767
unsigned int	16	0 a 65.535
signed int	16	O mesmo que **int**
short int	16	O mesmo que **int**
unsigned short int	16	0 a 65.535
signed short int	16	O mesmo que **short int**
long int	32	-2.147.483.647 a 2.147.483.647
signed long int	32	O mesmo que **long int**.
unsigned long int	32	0 a 4.294.967.295
float	32	Seis dígitos de precisão
double	64	Dez dígitos de precisão
long double	80	Dez dígitos de precisão

Definições de Variáveis

- Definições de Variáveis
  - <tipo> <nome>;
  - <tipo> <nomev1>,<nomev2>...

- Variáveis Globais X Locais
  - Obs: Variáveis locais devem estar na primeira parte de um bloco.

Constantes

- Definição de Constantes
  - #define <nome_constante> valor
  - Exemplo: #define PI 3,141592653
    #define ARQUIVO "arq1.ini"

•Obs: Não usa-se ";" para definir constantes.
•#define SOMA 100+120; // causa erro
•#define oct 0xFF // 255 decimal
•…
•x= SOMA; → x=100+120;; //erro

Operadores

- Atribuição
- Aritméticos
- Relacionais
- Lógicos
- Manipulação de Bits
- Assinalamento
- Endereço

Atribuição

- Operadores de Atribuição

```
a =10;
b = c * valor + getval(x);
a = b = c = 1;
```

Operadores Aritméticos

Operador	Ação
+	Soma
-	Subtração
*	Multiplicação
/	Divisão
%	Módulo
-	Negativo (Op. unário)

Operadores Relacionais

Operador	Ação
>	Maior
>=	Maior ou igual
<	Menor
<=	Menor ou igual
==	Igual
!=	Diferente (não igual)

Não existem operadores => , =< ou <>!!
Não Confunda = (atribuição) e == (comparação)!!

Operadores Lógicos

Operador	Ação
&&	AND
||	OR
!	NOT

Exemplos **Short circuit:**
(a==b)&&(b==c) // se a!=b não será avaliada a segunda condição
(a==b)||(b==c) // se a==b não será avaliada a segunda condição

Operadores de Bits

Operador	Ação
&	AND
|	OR
^	XOR
<<	Shift left
>>	Shift right
~	Not (complemento)

Observação: x<<2 // Rotaciona “x” 2 vezes à esquerda.

Operadores de Assinalamento

var = var **op** expr → var **op=** expr

Exemplos:

```
i += 2 ;  // equivale a : i=i+2;
j >>= 3; // equivale a : j = j>>3;
z &= flag; //equivale a : z = z & flag;
```

Operadores de Pré e Pós-Incremento

++i	i= i+1;
i++	i = i ; i= i +1;
z=++a	a = a +1 ; z = a;
z=a++	z = a ; a = a +1;

Observação: O mesmo raciocínio é válido para o sinal de subtração.

Operadores de Endereço

Operador	Ação
&	Endereço de uma variável
*	Conteúdo de uma variável

Exemplo:

```
int var , *x;
var = 10;
x = &var;
var = *x;  // var passa a conter o seu
                    endereço como valor
```

Outros Operadores

Operador	Ação
~	Not
sizeof()	Tamanho de uma variável

Precedências

Maior	( ) [ ] ->
	! ~ ++ -- - (tipo) * & sizeof
	* / %
	+ -
	<< >>
	< <= > >=
	== !=
	&
	^
	!
	&&
	! !
	?:
	= += -= *= /= etc.
Menor	,

Expressões

- Ordem de avaliação
  - Operadores, constantes e variáveis são elementos que constituem expressões em C
  - Em C, a ordem de avaliação de sub-expressões não é garantida
    - Ex.: x = f1( ) + f2 ( )

Conversões de Tipo

- Quando constantes e variáveis de tipos diferentes são misturadas em uma expressão, elas são convertidas para um mesmo tipo onde prevalecerá o maior.

Conversões de Tipo

SE um operando é **long double**
ENTÃO o segundo é convertido para **long double**
SENÃO, SE um operando é **double**
ENTÃO o segundo é convertido para **double**
SENÃO, SE um operando é **float**
ENTÃO o segundo é convertido para **float**
SENÃO, SE um operando é **unsigned long**
ENTÃO o segundo é convertido para **unsigned long**
SENÃO, SE um operando é **long**
ENTÃO o segundo é convertido para **long**
SENÃO, SE um operando é **unsigned int**
ENTÃO o segundo é convertido para **unsigned int**

Conversões de Tipo

- Cast
  - Força uma expressão a ser de um determinado tipo
    - Ex.: (float) x /2 ;

Conversões de Tipo

- Exemplo
- Exercitando o uso de cast, faça um programa que imprima o resultado da divisão por 2 dos números inteiros de 1 a 20.

Conversões de Tipo

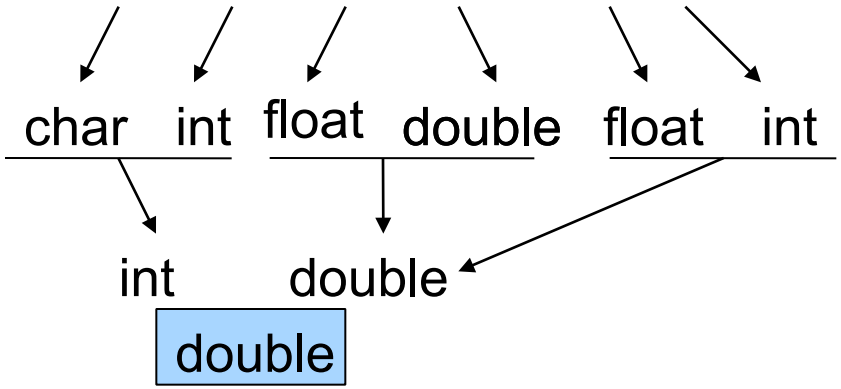
- Exemplo

```
#include <stdio.h>

main(){

    int i;

    for (i =1 ; i<=10; ++i ){
        printf("%d / 2 é : %f \n ", i , (float) i/2);
    }
}
```

Conversões de Tipo

Exemplo:

```
char ch;
int i;
float f;
double d;

Result = (ch / i) + (f *d) – (f + i);
```



Função Printf

- Printf();
  - Realiza saída para tela.

printf (“string de controle”, lista de argumentos);

Código	Significado
%c	Exibe Caractere
%d	Exibe inteiro na forma decimal
%e	Exibe número em notação científica
%f	Exibe um ponto flutuante em forma decimal
%s	Exibe uma String
%p	Exibe um ponteiro

* Existem outros tipos de parâmetros possíveis para utilização com printf();

Função Printf

- Outras opções
  - Exemplos:
    - %5d //imprime decimal com pelo menos 5 digitos
    - %05d //Imprime decimal com com pelo menos 5 digitos completando com 0,caso necessário.
    - %10.4f //exibirá float com 10 posições e no mínimo 4 casas decimais.
    - %5.7s //String com no mínimo 5 e máximo 7 posições
    - %-6.2f // O sinal de menos define alinhamento à esquerda

Função Printf

- Exemplos
  - printf("%-6.2f", 123.234);
    - Saída |123.23 |
  - printf("%06.2f", 3.234);
    - Saída |0003.23|

Função Scanf

- Scanf();
  - É uma das funções de entrada de dados da linguagem C. Lê dados do teclado

scanf (“string de controle”, lista de argumentos);

Código	Significado
%c	lê Caractere
%d	lê inteiro na forma decimal
%e	lê número em notação científica
%f	lê um ponto flutuante em forma decimal
%s	lê uma String
%p	lê um ponteiro

* Existem outros tipos de parâmetros possíveis para utilização com printf();

Função Scanf

- Exemplos
  - scanf(“%d,%d”, &a , &b); //entrada válida 10, 15
  - scanf(“%s” , nome); //Não precisa do “&”
  - scanf(“%d%d”, &r, &c); //aceita 10 20 , mas erro com 10,20.

Introdução às Funções

- Legibilidade
- Reutilização

- Forma Geral

```
<Tipo de Retorno>   <Nome da Função> (Argumentos)
{
        <Corpo da Função>
}
```

Introdução às Funções

```c
#include <stdio.h>
 int mensagem () /* Funcao simples: so imprime Ola! */
 {
        printf ("Ola! ");
        return(0);
 }

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

Introdução às Funções

- Declaração de Variáveis
- Argumentos

```
#include <stdio.h>
 int square(int x)
 {
    printf (“O quadrado é %d \n”, (x * x) );
    return(0);
 }

 int main ()
 {
    int valor = 2;
    square(valor);
    return(0);
 }
```

Introdução às Funções

- Retornando Valores

```
#include <stdio.h>
 int prod(int x , int y)
 {
    return(x * y);
 }

 int main ()
 {
    int valor1, valor2, produto;
    valor1 = 2;
    valor2 = 10;
    produto = prod(valor1, valor2);
    printf("O produto é: %d \n", produto);
    return(0);
 }
```

Introdução às Funções

- Lendo dados do teclado

```
#include <stdio.h>
int leNumero() /* Calcula o quadrado de x */
{
    int num;
    printf ("Entre com um numero: ");
    scanf ("%d",&num);
    printf ("\n\n");
    return(num);
}
int main ()
{
    int numeroLido = leNumero();
    printf (“Você digitou %d”, numeroLido);
    return(0);
}
```

Exercício

- Escreva um programa em C para ler um número através do teclado e imprimir o seu quadrado.

Comando de Seleção If

- Forma Geral

```
if (expressão) comando;
else comando;
```

Comando de Seleção If

- IFs Aninhados

```
if (expressão) comando;
else
        if (expressão) comando;
        else
                if (expressão) comando;
                else  ...
```

Comando de Seleção If

- IFs Aninhados

```
if (expressão)
        comando;
elseif (expressão)
        comando;
elseif (expressão)
        comando;
...
else
        comando;
```

Verdadeiro e Falso em C

- Verdadeiro
  - Qualquer valor diferente de zero
- Falso
  - Zero

Exercício

- Programa do Número Mágico
  - Escrava um programa que gera um número aleatório e interage com o usuário para saber se ele acertou o número.

Operador Ternário “?”

- Forma Geral

Expressão1 ? Expressão2 : Expressão3;

Ex.: Faça um programa para calcular raiz quadrada utilizando o operador ternário.

Comando de Repetição For

- O laço de repetição for

```
for (inicialização; condição ; incremento){
        comandos;
}
```

Comando de Repetição For

Exemplo:

```
#include <stdio.h>

void main(){

        int x;
        for (x=1; x<=100;x++){
                printf(“%d”, x);
        }
}
```

Comando de Repetição For

Exemplo 2:

```
#include <stdio.h>

void main(){

        int x, z;
        for (x=100; x!=65; x-=5){
                z = x*x;
                printf("O quadrado de %d, eh %f", x, z);
        }
}
```

Variações do Laço For

- Exemplo 1

```
#include <stdio.h>
void main(){
        int x,y;
        for(x=0,y=0; x+y<100; x++, y++){
                printf("%d",x+y);
        }
}
```

Variações do Laço For

- for (x =0, y=0 ; x+y <10 ; ++x) { <corpo do for>}
- for (i=0; i<100 && fim !='N' ; i++) { <corpo do for>}
- for(prompt();t=readnum();prompt()) { <corpo do for>}
- for(x=0; x!=10;) { <corpo do for>}

Variações do Laço For

- Laço Infinito
  - for(;;)
    - Interrompido somente com o break

Cláusula Break (no For)

- A cláusula break tem encerra o controle de um laço

```
#include <stdio.h>
void main(){
        int t;
        for(t=0; t<100; t++){
                printf("%d",t);
                if(t==0) break;
        }
}
```

Cláusula Continue

- Funciona de maneira semelhante ao break, mas tem como função forçar a próxima interação de um laço.

Exemplo:

```
#include <stdio.h>
void main(){
        int x;
        for(x=0; x<100; x++){
                if(x%2) continue;
                printf ("%d", x);
        }
}
```