Programação Estruturada Aula 2 - Introdução

Prof. Luis Carlos Retondaro

Técnico em Telecomunicações
2^o Ano

CEFET/RJ - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

Campus Petrópolis

2017

Sumário

1. Programação em C
2. Tipos de dados
3. Identificadores
4. Variáveis
5. Informações extras
6. Referências

O que é C?

- É uma linguagem de programação de computadores usada para criar um conjunto de instruções que o computador consiga executar;
- Sintaxe: São as regras detalhadas para cada construção de instrução válidas;
- O C possui palavras-chaves (ou reservadas) que combinadas com sua sintaxe formam a linguagem de programação em C;
- Todas as palavras-chaves em C são minúsculas e não podem ser usadas com nenhum outro propósito;
- Todo programa em C consiste em uma ou mais funções, porém a única que precisa estar presente é a main().

Forma geral de um programa em C

```
declarações globais

<tipo de retorno> f1(lista de parâmetros){
    sequência de comandos
}

<tipo de retorno> f2(lista de parâmetros){
    sequência de comandos
}

.
.
.
<tipo de retorno> fn(lista de parâmetros){
    sequência de comandos
}



<tipo de retorno> main(lista de parâmetros){
    sequência de comandos
}
```

A forma de um programa em C

Palavras reservadas em C

A linguagem C possui um total de 32 palavras conforme definido pelo padrão ANSI, que são elas:

• 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

Alguns compiladores incluem outras palavras reservadas como: asm, cdecl, far, fortran, huge, interrupt, near, pascal, typeof.

A forma de um programa em C

- As palavras-chaves combinadas com a sintaxe do C formam a linguagem de programação C;
- Muitos compiladores C acrescentaram diversas palavras-chaves para explorar melhor a organização de memória de alguns processadores;
- Todas as palavras-chaves em C são minúsculas e o C é *case sensitive*, ou seja, maiúsculas e minúsculas são diferentes.

Biblioteca do C

- Tecnicamente é possível criar um programa útil e funcional que consista apenas em comandos criados pelo programador, mas isso é raro;
- Alternativamente o C possui uma biblioteca ampla e a maioria dos programas inclui chamadas a funções contidas em sua biblioteca padrão;
- Todo compilador C vem com uma biblioteca C padrão de funções que realiza as tarefas necessárias mais comuns;
- Quando o compilador C não encontra no programa uma função que está sendo chamada, o nome da função é memorizada e o linkeditor combina o código desenvolvido com o código presente na biblioteca.

Compilação de um programa em C

- A compilação de um programa em C consiste em 3 passos:
  1. Criar o programa;
  2. Compilar o programa;
  3. Linkeditar o programa com as funções necessárias da biblioteca.
- Com o gcc instalado é simples compilar programas em c, se o programa consistir em um único arquivo, pode-se executar o comando no terminal:

```
gcc nomeProg.c -o nomeProg.e
```

- Para executar o programa pode-se digitar o comando:

```
./nomeProg.e
```

Primeiro programa em C

```c
#include <stdio.h>

//Primeiro programa
int main(){
  printf("Primeira aula de programação \n segundo semestre de 2014 \n");
  return(0);
}
```

- **#include <stdio.h>** - Inclui o arquivo stdio.h que possui declarações de funções para E/S de dados;
- **//Primeiro programa** - É um comentário de uma linha. Os comentários são importantes para auxiliar no entendimento/clareza do programa;
- **int main()** - Indica a declaração de uma função denominada main. Essa é a função principal e obrigatória pois é esta função que será chamada quando o programa for executado.

Primeiro programa em C

```c
#include <stdio.h>

//Primeiro programa
int main(){
  printf("Primeira aula de programação \n segundo semestre de 2014 \n");
  return(0);
}
```

- {} - O conteúdo de uma função é delimitado por chaves;
- O código que estiver entre o conjunto de chaves será executado sequencialmente quando o programa for executado;
- **int** - Significa o tipo do retorno da função;
- **return(0)** - Retorna zero ao SO indicando que o programa foi bem sucedido, outros números retornados no main indicam o código de uma condição de erro.

Primeiro programa em C

```c
#include <stdio.h>

//Primeiro programa
int main(){
   printf("Primeira aula de programação \n segundo semestre de 2014 \n");
   return(0);
}
```

- **printf()** - Recebe dados e imprime os dados como saída através do monitor;
- É por causa do uso da função printf() pelo programa que devemos incluir o arquivo-cabeçalho stdio.h;
- **\n** - comando de mudança de linha (*new line*);
- **;** - Os comandos do C terminam com um ponto e vírgula.

Mapa de memória do C

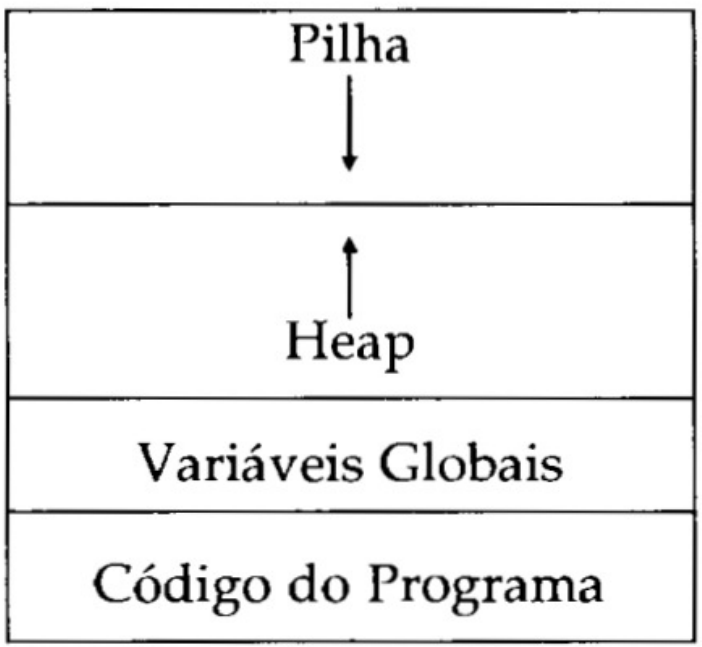
- Um programa em C compilado cria e usa 4 regiões logicamente distintas na memória, que possuem funções específicas;
- A primeira região é a memória que contém o código do seu programa;
- A segunda é onde as variáveis globais são armazenadas;
- As duas regiões restantes são a pilha e o heap.

Mapa de memória do C

- A pilha tem diversos usos durante a execução do programa, possui o endereço de retorno de chamadas à funções, argumentos para as funções e variáveis locais, guarda o estado atual da CPU;
- O heap é uma região de memória livre que o programa pode usar, através de funções de alocação dinâmica em C (aplicações como listas encadeadas e árvores);
- A disposição exata do programa na memória pode variar de compilador para compilador e de ambiente para ambiente.

Mapa de memória do C

- Disposição conceitual do programa na memória;



Informações extras: O que são erros de compilação

- Se o programa não estiver de acordo com as regras de sintaxe da linguagem ocorrerão erros de compilação;
- Le e entender esses erros é muito importante para o programador;

```
#include <stdio.h>
void main(){
 printf("Hello, world!\n");
```

Erro:

```
gcc hello.c -o hello.e
hello.c: In function main :
hello.c:5: error: syntax error at end of input
```

Informações extras: O que são erros de execução

- Quando o comportamento do programa diverge do esperado e pode acontecer mesmo quando o programa compila corretamente;

```
#include <stdio.h>
void main(){
  printf("Hello, world! $#%#@%\n");
}
```

```
gcc hello.c -o hello.e
Hello, world! $#%#@%
```

Informações extras: O que é um depurador

- Ferramenta que executa um programa passo a passo (*debugger*);
- Ajuda a encontrar erros de execução (bugs);
- **Exemplo:** GNU Debugger (GDB) e a maioria das IDEs possuem *debugger*.

Introdução

- Variáveis e constantes são os elementos básicos que um programa manipula;
- Uma variável é um espaço reservado na memória do computador para armazenar um tipo de dado determinado;
- Variáveis devem receber nomes para poderem ser referenciadas e modificadas quando necessário;
- Muitas linguagens de programação exigem que os programas contenham declarações que especifiquem o tipo e um valor inicial para as variáveis.

Tipos de dados

- Os tipos definem as propriedades dos dados manipulados em um programa, as variáveis são armazenadas de acordo com seus tipos (Informação);
- Existe 5 tipos básicos de dados em C:
  1. **int:** Usado para armazenamento de qualquer número inteiro negativo, nulo ou positivo (inteiro). **Exemplo:** -5, 0, 2;
  2. **float:** Usado para armazenar qualquer número real negativo, positivo ou nulo (ponto flutuante com precisão de 5 ou 6 dígitos significativos). **Exemplo:** -1.78, 0, 98.5, 2;
  3. **double:** Usado para armazenar qualquer número real negativo, positivo ou nulo (ponto flutuante de precisão dupla com precisão de 14 ou 15 dígitos significativos). **Exemplo:** -1.78, 0, 98.5, 2;

Tipos de dados

- Existe 5 tipos básicos de dados em C:

1. **char:** Usado para se armazenar quaisquer letras e números (conjunto de caracteres alfanuméricos) (caracter). **Exemplo:** 'A', 'H', '1';
2. **void:** Usado para declarar explicitamente uma função que não retorna valor algum ou para criar ponteiros genéricos. (sem valor). **Exemplo:**

```
void Imprime(){
   Conjunto de intruções
}
```

Tipos de dados

- O tamanho e a faixa desses tipos de dados variam de acordo com o tipo de processador e com a implementação do compilador C;
- Um caracter ocupa geralmente 1 byte e um inteiro 2 bytes, porém essa suposição não pode ser sempre feita se o programador deseja que seus programas sejam portáveis a uma gama mais ampla de computadores:
- O padrão ANSI estipula apenas a faixa mínima de cada tipo de dado, mas não o seu tamanho em bytes.

Tipos de dados em C

Tipos de dados básicos definidos no padrão ANSI.

Tipo	**Tamanho aprox. em bytes**	**Faixa**
char	1	-127 a 127
int	2	-32.767 a 32.767
float	4	3.4E-38 a 3.4E+38
double	8	1.7E-308 a 1.7E+308
void	0	Nenhum valor

Modificando os tipos básicos

- Exceto o **void**, os demais tipos básicos de dados podem ter vários modificadores precedendo-os;
- Um modificador é usado para alterar o significado de um tipo básico (adaptá-lo mais precisamente). Totalizam 4 modificadores:
  1. **signed:** Assume inteiros com sinal e sem sinal, porém é redundante dado que a declaração padrão assume números com sinal;
  2. **unsigned:** Usado para definir números inteiros sem sinal;
  3. **short:** Altera o tamanho do byte inteiro para um valor menor;
  4. **long:** Altera o tamanho do byte inteiro para um valor maior;

Modificando os tipos básicos

- Os modificadores signed, short, long e unsigned podem ser aplicados aos tipos básicos caracter e inteiro;
- O modificador long também pode ser aplicado a double;
- O padrão ANSI elimina o long float porque ele tem o mesmo significado de um double;
- O uso de signed com inteiros é permitido, porém é redundante porque a declaração padrão de inteiros assume números com sinal;
- O uso mais importante de signed é modificar char em implementações em que esse tipo padrão não tem sinal.

Modificando os tipos básicos

- Algumas implementações podem permitir que unsigned seja aplicado aos tipos de ponto flutuante (**unsigned double**), porém essa prática reduz a portabilidade do código e não é recomendável;
- Qualquer tipo expandido ou adicional não definido pelo padrão proposto ANSI provavelmente não será suportado por todas as implementações de C;
- A diferença entre inteiros com ou sem sinal é a forma como o bit mais significativo do inteiro é interpretado;
- Para representar um inteiro com sinal o compilador gera um código sendo que o bit de mais alta ordem é o indicador de sinal;
- Se o indicador de sinal é igual a 0 (zero) o número é positivo, se for igual a 1 o número é negativo.

Tipos de dados em C

Todos os tipos de dados definidos no padrão ANSI.

Tipo	**Tamanho aprox. em bytes**	**Faixa**
char	1	-127 a 127
unsigned char	1	0 a 255
signed char	1	-127 a 127
int	4	-32.767 a 32.767
unsigned int	4	0 a 65.535
signed int	4	-32.767 a 32.767
short int	2	-16.383 a 16.383
unsigned short int	2	0 a 32.767
signed short int	2	-16.383 a 16.383
long int	4	-2.147.483.647 a -2.147.483.647

Tipos de dados em C

Todos os tipos de dados definidos no padrão ANSI.

Tipo	**Tamanho aprox. em bits**	**Faixa**
signed long int	4	-2.147.483.647 a -2.147.483.647
unsigned long int	4	0 a 4.294.967.295
float	4	seis dígitos de precisão
double	8	dez dígitos de precisão
long double	10	dez dígitos de precisão

Exemplo de tamanho dos tipos de dados

```c
//Programa que imprime o tamanho de alguns tipos de dados
#include <stdio.h>

int main()
{
    int intValue = sizeof(int);
    int charValue = sizeof(char);
    int UshortIntValue = sizeof(unsigned short int);

    printf("char : %d bytes\n", charValue);
    printf("char : %ld bytes\n", sizeof(char));
    printf("int : %d bytes\n", intValue);
    printf("SO int : %ld bytes\n", sizeof(int));
    printf("U short int: %d bytes\n", UshortIntValue);
    printf("short int: %ld bytes\n", sizeof(short int));
    printf("long int: %ld bytes\n", sizeof(long int));
}
```

Exemplo de tamanho dos tipos de dados

Resultado da execução do programa

```
char : 1 bytes
char : 1 bytes
int : 4 bytes
SO int : 4 bytes
U short int: 2 bytes
short int: 2 bytes
long int: 8 bytes
```

Identificadores

- Em C, o nome de variáveis, funções, rótulos, etc são chamados de identificadores;
- O primeiro caracter de um identificador deve ser uma letra ou um underline, os demais devem ser letras, números ou underlines;
- Os demais caracteres não são aceitos!
- Exemplos:

Correto	**Incorreto**
count	1count
teste23	hi!here
high_balance	high...balance
childNumber	child number

Identificadores

- O padrão C ANSI determina que identificadores podem ter qualquer tamanho, mas pelo menos os 6 primeiros caracteres devem ser significativos se ele estiver envolvido em um processo de linkedição;
- Esses identificadores são chamados de nomes externos e incluem nomes de funções e variáveis globais que são compartilhadas entre arquivos;
- Se o identificador não é utilizado em um ambiente externo de linkedição os primeiros 31 caracteres serão significativos;
- A quantidade de caracteres significativos pode variar de compilador para compilador.

Identificadores

- Um identificador pode ser maior que o número de caracteres significativos reconhecidos pelo compilador, porém os que ultrapassarem o limite serão ignorados;
- É uma boa prática de programação que os identificadores possuam nomes significativos para auxiliar a compreensão do programa;
- Nomes de identificadores excessivamente longos são incomodos.

Identificadores

- O C é *case sensitive*, isto é maiúsculas e minúsculas fazem diferença. Exemplo:

```
int Soma, SOMA, SoMa, sOmA, soma;
```

- Um identificador não pode ser igual a uma palavra-chave de C, e não deve ter o mesmo nome que as funções criadas pelo usuário ou as existentes na biblioteca C.

Variáveis

- É uma posição nomeada de memória que é usada para guardar um valor que pode ser modificado pelo programa;
- Todas as variáveis em C devem ser declaradas antes de serem usadas;
- Uma declaração de variável em C consiste no nome de um tipo, seguido do nome da variável, seguido de ponto-e-vírgula;
- Forma geral de declaração:

```
tipo_da_variável lista_de_variáveis;
```

Variáveis

- Exemplo de declaração de variáveis:

```
int i, j, age;
short int number;
double balance, profit, loss;
float price;
```

- Em C usualmente são utilizados nomes de variáveis em minúsculo e constantes em maiúscula.

Variáveis

- Exemplo de utilização de variáveis:

```
#include <stdio.h>
// Exemplo de uso de variáveis do tipo char e int
int main()
{
   char Ch;
   int num;

   Ch='D';
   printf("%c", Ch);

   num=10;
   printf("%d", num);
   return 0;
}
```

Tabela ASCII

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0 16	Caracteres de Controle															
32		!	”	#	$	%	&	’	(	)	*	+	,	-	.	/
48	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
64	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y	Z	[	/	]	^	_
96	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
112	p	q	r	s	t	u	v	w	x	y	z	{	—	}	~	

Obtendo o tamanho de um tipo

- O comando sizeof(tipo) retorna o tamanho, em bytes, de um determinado tipo;
- **Exemplo:**

```
printf("%d \n", sizeof(int));
```

Referências

1. C completo e Total. 3a ed. Revista e Atualizada, Hebert schildt, Makron Books Ltda.
2. Linguagem C. DAMAS, Luis. 10a. Edição. LTC, 2014.
3. Aprendendo a programar, programando na linguagem C. 3a. Edição. Edição digital, Jaime Evaristo.
4. Notas de aula prof. Alexandre Xavier Falcão, UNICAMP, 2005.