Linguagem C – Princípios Básicos (parte 1)

Objetivos

O principal objetivo deste artigo é explicar alguns conceitos fundamentais de programação em C. No final será implementado um programa envolvendo todos os assuntos abordados. Todos os exemplos mostrados no decorrer neste tutorial podem ser testados em qualquer compilador C/C++, pois obedecem ao padrão C ANSI.

Introdução

A linguagem C foi desenvolvida por Dennis Ritchie em 1972 para ser utilizado com o sistema operacional UNIX. Desde então não parou de crescer, sendo até hoje usada por diversos programadores. O seu sucesso se deve a grande flexibilidade que ela oferece ao programador. Existem diversas vantagens em se utilizar C, vejamos algumas delas: possui um conjunto compacto de palavras-chaves e de tipos de dados evitando varias operações desnecessárias usam de ponteiros que permite o acesso de baixo nível a memória, os parâmetros das funções não são passados através de referencia embora isto seja possível fazendo uso de ponteiros. A linguagem C é considerada de baixo nível ou de nível médio, o que significa obter um melhor controle do hardware podendo assim manipular bits, bytes e endereços.

Devido ao grande numero de programadores que utilizam a linguagem C existe uma vasta gama de compiladores C/C++ e bibliotecas disponível no mercado. Sendo alguns deles gratuitos, por exemplo, o dev-c++ para Windows e o gcc para Linux. Para que todos esses compiladores de diferentes fabricantes possam ser compatíveis entre si existe o padrão C que foi estabelecido pelo comitê criado em 1983 pela ANSI (American National Standards Institute).

Definição de Compiladores

Os compiladores são programas sofisticados que traduz o código fonte para uma linguagem que o computador possa entender (linguagem de maquina). O compilador lê as instruções uma por vez, verificando sua sintaxe e convertendo a instrução para a linguagem de maquina, porém não a executa ainda. Este processo é repetido até que a ultima instrução seja lida e convertida para linguagem de maquina. Caso não haja erro o compilador gera um programa em disco com o sufixo. OBJ contendo as instruções traduzidas. Este programa só será executado quando todas as rotinas necessárias para execução estiverem agregadas. Este processo é feito pelo “linkeditor” que, além disso, cria um programa com o sufixo.EXE que pode ser executado diretamente pelo sistema operacional.

Palavras Chaves

A tabela a seguir apresenta as **32 palavras-chaves existentes em C** e definidas como padrão **ANSI.** Alguns compiladores podem fazer uso de outras palavras-chaves (palavras reservadas) que não estão presentes na lista abaixo. As palavras-chaves são escritas sempre em letras minúsculas.

Tabela 1 – Lista das palavras-chaves

Palavras chaves em C (padrão ANSI)			
auto	Double	int	Struct
break	Else	long	Switch
case	Enum	register	typedef
char	Extern	return	union
const	Float	short	unsigned
continue	For	signed	void
efault	Goto	sizeof	volatile
do	If	static	while

Estrutura básica de um programa em C

Um programa é composto de uma ou mais funções. Sendo que a única função obrigatória é a **main ()**. Esta é a primeira função a ser chamada toda vez em que o programa é executado. Toda função deve ter o seu nome precedido de parênteses **"()"**, indicando que se trata de uma função. Os símbolos **"{" e"}"** representam o inicio e o termino da função respectivamente. O programa abaixo mostra à estrutura básica de um programa escrito em C. A palavra reservada **void** na frente da função **main ()** indica que ela não retorna nenhum valor.

```
#include<stdio.h>

/* Programa que utiliza os operadores aritméticos*/

void main()
{

   int x,y,soma,sub;
   float modulo,div,mult;
   x=69;
   y=24;

   //utilizando os operadores aritmeticos:
   soma=x+y;
   sub=x-y;
   mult=(float) x*y;
   div=(float) x/y; /*aqui é utilizado além do operador aritmetico de divisão
                      o operador Cast.*/
   modulo=(x % y);

   printf("%d + %d=%d\n",x,y,soma);  //imprime a soma dos dois números
   printf("%d - %d=%d\n",x,y,sub);   //imprime a subtração
   printf("%d x %d=%f\n",x,y,mult); //imprime a multiplicação
   printf("%d / %d=%f\n",x,y,div); //imprime a divisão
   printf("%d mod %d=%f\n",x,y,modulo);  //imprime o resto da divisão

}
```

Observando o programa acima podemos perceber que existem duas formas de comentá-lo. O primeiro é utilizando duas barras "//" que comenta todo o código contido naquela linha e não há necessidade de fechá-la. O segundo é usando os caracteres /* e */ que comentam todas as instruções localizadas entre eles. Quando comentamos uma instrução estamos na realidade impedindo que elas sejam compiladas.

Tipos de dados

Existem quatro tipos de dados básicos são eles: caractere, inteiro, ponto flutuante e ponto flutuante de precisão dupla (char, int, float e double). Na tabela 2 temos a descrição do tamanho e a faixa de abrangência dos tipos de dados básicos. Esses dados podem variar conforme o tipo de processador e a implementação do compilador C utilizado, mas este valor sempre será igual ou maior do que os mostrados na tabela.

Tabela 2 – Tipos de dados básicos

Tipos de dados básicos (padrão ANSI)			
Tipo	**Tamanho aproximado**		**Faixa de Abrangência**
	Bytes	**Bits**	
char	1	8	[-128... 127]
int	2	16	[-32768... 32767]
float	4	32	[3.4E-38... 3.4E38]
double	8	64	[1.7E-308... 1.7E308]

Podemos aplicar modificadores em dados do tipo caractere (char) e inteiro (int). No caso do double podemos aplicar somente o long e ao tipo float não são aplicados nenhum deles. O modificador altera o significado de um tipo de dado básico. Os modificadores existentes são:

•Long: É empregado aos tipos de dados int e double para indicar um tipo de dado maior. Também pode ser declarado em conjunto com o float, mas isto seria a mesma coisa de usar o tipo double.

•Short: Muito utilizado para indicar valores inteiros curtos (short int). Normalmente possui o mesmo tamanho do tipo int, mas são armazenados em um numero menor de bytes.

•Signed: Este modificador é usado em dados do tipo caractere (char) para permitir o uso de sinal. Também pode ser usado com inteiros, porém isto seria redundante, pois sua declaração padrão já permite um número com sinal.

•Unsigned: Não permite o uso de sinais. Este modificador não é indicado para o tipo int, pois podem ocasionar algumas complicações caso seja atribuído em algum lugar do código um valor negativo.

Definição de Variáveis

Variáveis são posições da memória previamente identificadas para armazenar as informações a serem utilizadas pelo programa. Dependendo do local onde essas duas variáveis são declaradas podemos chamá-las de Locais ou Globais.

Variáveis locais: São aquelas declaradas dentro de funções. Essas variáveis só podem ser utilizadas pelas instruções que estão dentro do bloco em que foram declaradas. Um bloco de código inicia-se em abre-chaves ({) e termina em fecha-chaves (}). Essas variáveis locais só existem no momento em que o bloco de instrução que as contem está sendo executado.

Variáveis Globais: As declarações das variáveis globais são feitas fora de todas as funções do programa. As variáveis globais ao contrario das locais podem ser acessada de qualquer parte do programa e são guardas até o termino da execução do programa.

Nomes das variáveis

Os nomes da variáveis sejam elas locais ou globais identificação deve seguir as seguintes regras:

•O nome da variável deve conter um ou mais caracteres;

•O primeiro caractere sempre deve ser uma letra;

•Os caracteres subseqüentes podem ser letras, números ou o caractere "_" que geralmente é usado para indicar a separação entre duas palavras;

•O nome da variável ou identificador não pode ser igual às palavras-chaves já apresentadas, nem ter o mesmo nome de funções determinadas pelo usuário ou iguais as funções localizadas na biblioteca C.

Para entender melhor veja alguns exemplos de nomes de variáveis corretas e incorretas.

Correto:	**Incorreto:**
Soma1	1soma
soma	soma!
área_ triangulo	área... triangulo

A linguagem C é case sensitive, ou seja, as letras maiúsculas se diferem das minúsculas. Desta formas as variáveis soma, SOMA e Soma são distintas.

Declaração e inicialização de variáveis

Todas as variáveis precisam ser declaradas antes de serem utilizadas através da instrução:

<Tipo de dados> Nome_variável;

Toda instrução em C deve estar acompanhada de ";". O tipo de dados deve ser um dos mostrados na tabela 2 e o nome das variáveis deve seguir as regras citadas anteriormente.

A atribuição de valores para as variáveis é feita utilizando o sinal de igual "=" e obedece a seguinte sintaxe:

Nome_da_variavel = expressão;

A expressão pode ser uma simples constante ou uma formula matemática. Veja no exemplo abaixo como são feitas as declarações de variáveis e as atribuições de valores.

```
#include<stdio.h>

/* Programa que utiliza os operadores aritméticos*/

void main()
{

   int x,y,soma,sub;
   float modulo,div,mult;
   x=69;
   y=24;

   //utilizando os operadores aritmeticos:
   soma=x+y;
   sub=x-y;
   mult=(float) x*y;
   div=(float) x/y; /*aqui é utilizado além do operador aritmetico de divisão
                      o operador Cast.*/
   modulo=(x % y);

   printf("%d + %d=%d\n",x,y,soma);  //imprime a soma dos dois números
   printf("%d - %d=%d\n",x,y,sub);   //imprime a subtração
   printf("%d x %d=%f\n",x,y,mult); //imprime a multiplicação
   printf("%d / %d=%f\n",x,y,div); //imprime a divisão
   printf("%d mod %d=%f\n",x,y,modulo);  //imprime o resto da divisão

}
```

A saída do programa:

str = a

valor de x: 567

x = 789.564331 e xx = 8912.781553

Implementação de todos os conceitos abordados

```
#include<stdio.h>

/* Programa que utiliza os operadores aritméticos*/

void main()
{

   int x,y,soma,sub;
   float modulo,div,mult;
   x=69;
   y=24;

   //utilizando os operadores aritmeticos:
   soma=x+y;
   sub=x-y;
   mult=(float) x*y;
   div=(float) x/y; /*aqui é utilizado além do operador aritmetico de divisão
                      o operador Cast.*/
   modulo=(x % y);

   printf("%d + %d=%d\n",x,y,soma);  //imprime a soma dos dois números
   printf("%d - %d=%d\n",x,y,sub);   //imprime a subtração
   printf("%d x %d=%f\n",x,y,mult); //imprime a multiplicação
   printf("%d / %d=%f\n",x,y,div); //imprime a divisão
   printf("%d mod %d=%f\n",x,y,modulo);  //imprime o resto da divisão

}
```

Saída do programa:

Comprimento da transferência: 125,6

O programa acima utiliza a função **printf ()** para imprimir na tela o valor das variáveis. Para que esta função seja utilizada é necessário incluir a biblioteca padrão de entrada e saída de dados **(stdio. h)** através da diretiva **include**. Esta parte pode ser identificada no inicio dos programas mostrados **(“#include<stdio. h>”).** Esta biblioteca será descrita com mais detalhes em um próximo artigo.

A sintaxe da **função printf ()** utilizada é:

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

Na expressão de controles são colocados os caracteres a serem exibidos na tela e os códigos de formatação que indicam o formato em que os argumentos devem ser impressos. A função **printf ()** pode conter um ou mais argumentos. Caso haja mais de um eles devem ser separados por vírgula. O caractere \n pula para linha de baixo.

Conclusão

Neste artigo aprendemos quais são os tipos de dados existentes, a definição de variáveis locais e globais, como elas são declaradas e inicializadas. No próximo artigo serão abordados os conceitos de operadores