FIEP SESI SENAI IEL

Lógica de programação

Linguagem C

Kleber Petry
kleber.petry@pr.senai.br

Conceito da linguagem C

A linguagem de programação C é uma linguagem voltada para programação estruturada, de propósito geral, que tem como principais características:

- **Portabilidade**: é possível criar um programa na linguagem C e, com poucas adaptações, compilar em outra plataforma.
- **Modularidade**: o C permite usar apenas os recursos que serão necessários para nosso programa e também para programar de maneira que facilita o reuso de código.
- **Confiabilidade:** a linguagem C possui diversos compiladores e bibliotecas que nos permitem agregar confiabilidade a plataforma de programação.
- **Linguagem de nível médio:** a linguagem C possibilita acessar e fazer uso de recursos do hardware de maneira mais livre, sem ter a sintaxe complexa e especifica do *Assembly*.
- **Case sensitive (sensível ao caso):** na linguagem C, as letras minúsculas e maiúsculas são consideradas distintas, ou seja, se uma palavre reservada é definida com letras minúsculas (int, char, float) seu uso com letras maiúsculas não será reconhecido.

Tipos Básicos de dados em C

Tipo	Tamanho (bytes)	Limites
Int	4	-2.147.483.648 a 2.147.483.647
Unsigned int	4	0 a 4.294.967.295
Short int	2	-32.768 a 32.767
Unsigned int	2	0 a 65.535
Float	4	Seis dígitos de precisão
Double	8	Dez dígitos de precisão
Long double	12	Dez dígitos de precisão
Char	1	-127 a 127
Unsigned char	1	0 a 255

Declarando variáveis e constantes

Código	Descrição
int x;	Cria variável do tipo *int* (inteiro) chamada *x.*
int a, b1, c2;	Cria três variáveis inteiras chamadas a,b1 e c2.
float f;	Cria variável do tipo *float* (ponto flutuante) cahamda *f*.
double d;	Cria variável do tipo *double* chamada *d.*
long double y;	Cria variável do tipo *long double* chamada *y.*
char letra, ch;	Cria duas variáveis do tipo *char* chamadas *ch* e *letra*
const float PI=3.14;	Cria uma constante chamada de *PI* e inicializa com valor *3.14*

Operadores Básicos

	Descrição	Exemplo
=	Operador de atribuição. A variável da esquerda recebe o valor da expressão à direita	x=10; a=x;
+	Soma	b1=a+x; bi=a+2;
*	Multiplicação	c2=x*a; c2=100*b1;
/	Divisão	f=x/10.0;
-	Subtração	x=a-b1;
%	Resto da divisão(apenas para valores inteiros)	c2=a%b1; x=a%2;

++	Operador de incremento. Incrementa a variável em uma unidade. No exemplo *a=x++,* o incremento acontecerá depois da atribuição. A variável *a* receberá o valor antigo de *x*	x++;
		++x;
		a=x++;

--	Operador de decremento. decrementa a variável em uma unidade. No exemplo *a=--x,* o incremento acontecerá depois da atribuição. A variável *a* receberá o valor novo de *x*	x--; --x; a=--x;
&	Indica o endereço de memória ocupado por uma variável. *&x* mostrará o endereço da variável de *x*	&x

Exemplo de uso dos operadores

Exemplo	Significado
a + = 10;	a = a + 10;
a * = 2;	a = a * 2;
a / = 5;	a = a / 5;
a % = 2;	a = a % 2;

a = b++;	a = b; //atribuição primeiro
	b = b + 1;

a = ++b;	b = b + 1; a = b; //atribuição depois
a = b--;	a = b; b = b-1;
a = --b;	b = b-1; a = b;

Chamada de funções

Na linguagem C, temos diversas bibliotecas que nos fornecem funções para elaborarmos nossos programas. Cada função pode receber parâmetros e pode devolver um valor como retorno. Para realizarmos uma chamada de uma função, temos que incluir, no inicio do arquivo, a instrução, a instrução '*#include* <nome_biblioteca.h>' e realizar a chamada da função pelo nome.

Função **printf**

A função *printf* serve para realizar a escrita de dados na tela do computador, conforme veremos em nosso primeiro exemplo, em que usamos a função *printf* para escrever a mensagem **“meu primeiro programa em C”**

Principais códigos de controle função **printf**

Código de controle	Descrição
%d	Indica posição de um valor inteiro na notação decimal.
%o	Indica posição de um valor inteiro na notação octal.
%x	Indica posição de um valor inteiro na notação hexadecimal.
%c	Indica a posição de um caracter. Apenas um único caracter.
%s	Indica a posição de uma sequencia de caracteres(string).
%f	Indica posição de um valor ral na notação decimal.
%%	Insere um sinal de porcentagem.
\n	Insere uma nova linha.
\t	Insere uma tabulação.
\”	Insere aspas duplas (“).
\\	Insere contrabarra (\).

Função **scanf**

A função *scanf* permite realizar a leitura de dados digitados pelo usuário através do teclado. Funciona semelhante ao comando **leia**, que vimos nos português estruturado, porém com maiores possibilidades.

Função de leitura **scanf**

Código	Descrição
scanf (“%d %d”, &nota1, &nota2);	Realiza a leitura de dois valores inteiros separados por um espaço, atribuindo o primeiro valor no endereço da variável *nota1* e o segundo valor para *nota2.*
scanf (“%f”, &var);	Realiza a leitura de uma variável do tipo *float* e a armazena no endereço de memoria variável *var*.
scanf (“%c”, &ch);	Realiza a leitura de uma variável do tipo *char* e a armazena no endereço de memoria da variável *ch*.

ATENÇÃO: Se você esquecer de colocar o operador & junto da variável no comando *scanf*, o compilador não acusará erro, mas o valor não será armazenado no local desejado. Este é um erro bastante comum.

Funções Matemáticas

As bibliotecas-padrão da linguagem C dispõem de diversas funções matemáticas prontas como, por exemplo, potência e raiz quadrada. Para usar essas funções, você deve acrescentar o comando *#include <math.h>* juntamente com os outros comandos *#incude* no topo do programa.

Exemplos de funções matemáticas

Função	Descrição

double sqrt(double x);	Devolve a raiz quadrada do parâmetro.
	Exemplo:
	float f = sqrt(9);
	//calcula a raiz de 9

double pow(double x, double y);	Calcula a potência. O primeiro número passado como parâmetro elevado ao segundo numero passado como parâmetro. Exemplo: float f = pow(2,3) //Calcula dois elevado ao cubo

Comandos **if ..else**

Conforme vimos no português estruturado temos os comandos se..entao..senão. Na linguagem C, teremos os comandos *if* e *else*, que são equivalentes ao **se..então..senão** – porém, a linguagem C não faz uso de um termo especifico para marca o **então**. Os comandos de seleção servem para condicionar a execução de um trecho de código a uma determinada situação. Para especificarmos essas condições, precisaremos dos operadores relacionais e dos operadores lógicos da linguagem C.

Operadores relacionais

Operador	Tipo	Exemplo
>	Maior que	a>0
<	Menor que	a<0
>=	Maior ou igual a	a>=0
<=	Menor ou igual a	a<=10
!=	Diferente	a!=11
==	Igual	a==0

Operadores lógicos

Operador	Tipo	Exemplo
&&	E (Conjunção)	(a==0) && (b==1)
| |	OU (conjunção)	(A>0) | | (A==10)
!	NÃO (negação)	!(A==0)

Sintaxe do comando IF

	Sintaxe	Descrição

if Simples	if (condição){
	comando1;
	comando2;
	comando3;
	}
		Se a condição for verdadeira, executará os três comandos. Caso contrário, o programa continuará após o fecha-chaves.

if simples com um comando	if (condição) comando;	No caso de um único comando, é possível omitir as chaves.

Sintaxe do comando *if* com *else*	if (condição){
	comando1;
	comando2;
	}else{
	comando3;
	comando4;
	}
		Se a condição for verdadeira, executará os comando 1 e 2. Caso contrário, o programa executará os comandos 3 e 4.

if com *else* com um comando	if (condição)
	comando1;
	else
	comando2;
		No caso de um único comando é possível omitir as chaves.

Comandos switch...case

- O comando switch permite testar o valor de uma variável do tipo *char* ou *int* contra um sequência de valores.
- A sequência de comando que estiver associado ao valor encontrado na variável é executada.
- Caso não seja encontrada, é executada a sequencia de comandos definida em um bloco-padrão (default)
- O comando *switch* da linguagem C é equivalente ao comando *caso..seja* do português estruturado.

Comandos switch...case

- Um das principais diferenças entre o *caso..seja* do português estruturado e o *switch..case* é que, na linguagem C, não é permitido incluir intervalo de valores (por exemplo, 1...10)
- Outra diferença é que o bloco de comandos é finalizado pela palavra **break.** Caso não seja encontrada, os comandos subsequentes são executados, mesmo se pertencerem a outro bloco

```
Switch(variável){
Case valor1:
        sequencia de comandos;
        break;
Case valor2:
        sequencia de comandos;
        break;
Default:
        sequência de comandos;
}
```

Comandos de repetição

- Os comandos de repetição possibilitam repetir determinados trechos de código para solução de problemas que exijam repetição dos processos, facilitando a manutenção do código e a legibilidade.
- Na linguagem C, temos os comandos de repetição *for*, que seria equivalente ao **para..ate..faça** do português estruturado, o comando *while*, que seria equivalente ao **enquanto..faça** e, finalmente, o comando *do..while*, que seria semelhante ao **repita..ate**.
- É importante ressaltar que qualquer lçao ou bloco de repetição pode ser feito com qualquer um dos três comandos de repetição, com pequenas adaptações , assim, onde se pode usar um, pode-se usar outro.

Comandos de repetição

SINTAXE	EXEMPLO

for(inicialização;condição;incremento){
comando1;
comando2;
}
	int i;
	for(i=0;i<10;i++){
	printf(“%d\n”,i++);
	}

while(condição){
comando1;
comando2;
}
	int i=0;
	while(i<10){
	printf(“%d\n”,i++);
	}

do{
comando1;
comando2;
}while(condição);
	int i=0;
	do{
	printf(“%d\n”,i++);
	}while(i<10);

Comandos de repetição

- Alguns aspectos devem ser observados:

Em qualquer uma das três estruturas de repetição é possível omitir o abre-chaves e fecha-chaves quando o comando a ser repetido é apenas um.

Mas é recomendável usar sempre os delimitadores de bloco para aumentar a legibilidade do código.

A principal diferença entre o *while* e *do..while* é que, no comando *while*, a condição é testada antes do bloco ser executado, enquanto que, no comando *do..while,* o teste é realizado depois de executar o bloco.

No comando *for*, podemos colocar valores diferentes para o incremento como, por exemplo, *i=i+2*(para passo de 2 em 2)

Comandos de repetição

Note que, quando usamos como condição *i<10*, o próprio 10 não está incluído. Se quisermos incluí-lo, devemos usa *i<=10*

Dentro dos laços, podemos usar os mesmos comandos que fora da estruturade repetição, como os comandos *if..eles, swhitch..case*

```
int main(){
        int i;
        for(i=9;i>=0;i--){
                printf(“%d/n”,10-i);
        }
        return (EXIT_SUCCESS);
}
```

Exercícios

Praticar para aprender