Disciplina: Linguagem C
Professora: Sandra Hedler de Amorim
E-mail: sandra-famorim@educar.rs.gov.br

Linguagem de Programação C

Introdução - Conceitos Básicos

A linguagem C foi criada por Dennis Rotchie, implementada em UNIX.
C é considerada uma linguagem de médio nível, podendo trabalhar com bits, bytes e endereços para que possamos programar o hardware, ou seja, também pode ser considerada uma linguagem de baixo nível.
Código escritos em C são altamente portáteis, o que significa, que um programa criado em Windows pode ser executado em qualquer outro sistema operacional.
O C é uma linguagem estruturada, ou seja, um programa pode ser dividido em varias subrotinas(pequenos programas dentro de um programa maior), onde cada subrotina cria suas próprias variáveis e executa determinada função. Essas subrotinas somente serão executadas quando forem feitas suas chamadas no programa principal.

Compilação do Programa em C

1º - Criar o programa (código fonte).
2º - Compilar o código fonte (gerar o código objeto).
3º - Executar o código objeto.

Primeiro passo é necessário ter um compilador, onde vamos criar o programa (código fonte). A partir desse código fonte compilamos o mesmo para gerar um código objeto, ou seja, aquele código que criamos e podemos interpretar, entender, o programa não consegue entender, então ao compilar esse código fonte gera um código objeto o qual o computador consegue interpretar e executar. Após o objeto gerado, pode-se executar esse programa. Significa que não vamos executar o código fonte, primeiro vamos compilar o código fonte e gerar o código objeto para então executar esse código objeto. Com isso permite o executável pode ser colocado em máquina em máquina para sua execução, não sendo obrigatório o código fonte junto a ele.

Implementação

A implementação de um programa é a tradução de um algoritmo para a linguagem de programação que se deseja programar. O algoritmo sendo implementado, traduzido para uma linguagem de programação.

Estrutura de um programa C

```
#include <stdio.h>

main()
{
        bloco de comandos;
}
```

Devemos sempre levar em consideração, que diferente do que fizemos português estruturado, em C a primeira coisa que devemos fazer é a importação de bibliotecas que se dá a partir do #include e após a inclusão das bibliotecas.
No exemplo a seguir temos a biblioteca padrão de entrada e saída **stdio.h** seu nome vem da expressão inglesa standard input-output header . É a biblioteca que usamos para fazer toda a interação homem máquina, entrada e saída de dados. Essa biblioteca que tem algumas funções, por exemplo, a função printf() usada para imprimir dados na tela e scanf() usada para capturar dados do usuário.
Após temos a função principal, a função main() que delimita o bloco de comando por chaves. E todas as instruções que estão entre chaves deveram ser executadas para solucionar um determinado problema que vai solucionar e executado por um computador. Além da função principal main(), vamos criar outras funções que só serão executada a partir de chamadas nessa função principal.

Análise e Solução de Problemas

Primeiramente devemos especificar claramente o problema.
O que deve ser resolvido para solucionar o problema?
Verificado e encontrado essa questão passamos para o próximo passo.
Quais os dados e tipos de dados meu programa deve receber (entrada de dados)? Ou seja, quais são as informações que devem vir do mundo externo para que possa armazenar em memória para ser utilizadas posteriormente.
Qual a saída e o tipo de dado que o programa deve enviar (saída de dados)? Ou seja, as resposta que o programa deve fornecer ao final do problema proposto.

Após analisar os dados de entrada e saída, devemos descobrir uma solução, criar um algoritmo para a resolução do problema.
Escrever a solução em uma linguagem de programação e por último testar a solução.
Seguir sempre nessa ordem para solucionar e implementar em uma linguagem de programação.

Sintaxe e Semântica

Toda linguagem de programação possui uma sintaxe, ou seja, a gramatica que utilizo dos comandos que serão utilizados nesta linguagem. E a semântica é a lógica utilizada pelos programadores para solucionar o problema proposto.

A princípio o erro sintático o compilador acusa para o programador. Pois só gera o código objeto quando não encontrar nenhum erro sintático.
O erro semântico é o mais difícil de encontrar, pois é um erro da lógica utilizada pelo programador e a ferramenta não acusa esses erros e só vamos descobrir na execução, visualização do programa.
Devemos lembrar que sintaxe sempre tem que fazer da forma que está descrita ou estamos fazendo errado.

Uso da Informação

Todo trabalho realizado por um computador é baseado na manipulação das informações contidas na sua memória. Estas informações podem ser classificadas em dois tipos:
Instruções
Comandos de funcionamento da máquina e determinam a maneira como devem ser tratados os dados.
Dados
Correspondem à porção das informações a serem processadas pelas linhas de instruções.

Tipos de dados que podemos utilizar:
Constante: Toda informação que não sofrem alterações no decorrer do tempo de execução do programa.
Exemplo:

```
PI = 3.14
```

Variáveis: Há possibilidade de serem alteradas em algum instante no decorrer do tempo.
Exemplo:

temperatura, saldoBancario, idade

Expressões

Expressões são combinações de variáveis, constantes e operadores. Quando montamos expressões temos que levar em consideração a ordem com que os operadores são executados.

Expressões Aritméticas:

São aquelas cujo resultado da avaliação é do tipo numérico, seja ele inteiro ou real. Somente o uso de operadores aritméticos e variáveis numéricas são permitidos em expressões deste tipo.
Exemplo

A + B * C

Expressões Lógicas:
É aquelas cujo resultado da avaliação é um valor lógico (verdadeiro ou falso).
Exemplo

A > (B + C)

Tipos de Dados

Entre os tipos de dados que manipulamos temos os tipos numéricos e caracteres.
Int - tipo inteiro, trabalha com 32 bits que são 4 bytes.
float - tipo real, trabalha com 32 bits que são 4 bytes.
char - que representa 1 caractere, trabalha com 8 bits que é 1 bytes
double - tipo real, trabalha com 64 bits, 8 bytes.

Variáveis

- Correspondem a uma posição de memória;
- Conteúdo pode variar ao longo do tempo;
- Armazenam um só valor por vez;
- Armazenam valores de um só tipo (logo, tem tipo definido);
- Devem ser declaradas antes de serem utilizadas;
- Valor inicial imprevisível (posição de memória).

A forma geral de declaração de variável é:
Sintaxe:

tipo lista_de_variáveis;

Exemplo

```
int idade, matricula;
char nome, turma;
int idade = 90, matricula = 12356;
char letra = "A";
```

Identificadores de variáveis em C

- O primeiro caractere deve ser uma letra ou (sublinha) e as demais podem ser letras, números ou caracteres;
- Em C os primeiros 32 caracteres são significativos, ou seja, não adianta colocar identificadores muito extensos porque serão ignorados pelo compilador;
- O C distingue letras maiúsculas de minúsculas, ou seja, é case sensitive.
- Não devem ser utilizadas as palavras reservadas da linguagem, assim como não deve ser dado nomes a variáveis iguais os identificadores de funções.

Constantes

As constantes se caracterizam por valores fixos que não podem ser alterados.
Sintaxe:

```
#define nome_da_constante valor
```

ATENÇÃO: Note que não vai ponto e vírgula ao final.

Exemplo

```
#define PI 3.1415992
#define AREA 0.0
```

Atribuição de Dados

A operação de atribuição específica que a uma variável será dado um valor ou o resultado de uma expressão. O operador de atribuição é o sinal de igualdade (=).

Sintaxe:

```
variável = expressão;
```

Exemplo

```
a = 5;
b = 9.5;
```

a = b = c = 10;

Que se lê:

a recebe 5 (Valor 5 sendo atribuído a variável a)

b recebe 9.5

a recebendo b que recebe c que recebe 10. Ou seja, as 3 variáveis estão sendo inicializadas com valor 10.

O valor da expressão atribuído a variável tem que ser compatível conforme foi declarada.

Operadores Aritméticos

Os operadores aritméticos são usados para desenvolver operações matemáticas.

Operador	Ação
+	Soma (inteira e ponto flutuante)
-	Subtração ou troca de sinal (inteira e ponto flutuante)
*****	Multiplicação (inteira e ponto flutuante)
/	Divisão (inteira e ponto flutuante)
%	Resto de divisão (de inteiros)
++	Incremento (inteiro e ponto flutuante)
--	Decremento (inteiro e ponto flutuante)

Operadores Relacionais

Operadores relacionais existem para estabelecer uma relação entre dois elementos, cujo resultado da comparação será sempre igual a 0 (falso) ou 1 (verdadeiro).

Operador	Ação
>	Maior do que
>=	Maior ou igual a
<	Menor do que
<=	Menor ou igual a
==	Igual a
!=	Diferente de

Atenção: Os operadores >=, <=, == e **!=** tem que ser digitados lado a lado sem espaço entre eles.

Operadores Lógicos

Para fazer operações com valores lógicos (1 - verdadeiro e 0 - falso) temos os operadores lógicos. Iguais aos operadores relacionais, também retornam um valor final lógico (verdadeiro ou falso). Mas não servem para comparar resultados de expressões ou números. Apenas para comparar outros valores lógicos.

Operador	Ação
&&	AND(E)
||	OR(OU)
!	NOT(NÃO)

Usando os operadores relacionais e lógicos podemos realizar uma grande gama de testes. A tabela-verdade destes operadores é dada a seguir:

Negação	
A	**!A**
0	1
1	0

Conjunção		
A	**B**	**(A && B)**
0	0	0
0	1	0
1	0	0
1	1	1

Disjunção		
A	**B**	**(A || B)**
0	0	0
0	1	1
1	0	1
1	1	1

O operador de negação é unário, pode-se negar uma condição apenas. Quando trabalhamos com a negação se inverte o estado da condição.
As operações de conjunção e disjunção são operadores binários, em que trabalham com duas ou mais condições.
Na conjunção somente tem a saída igual a 1, se ambas as condições forem iguais a 1(verdadeiras). Basta uma ser igual a 0 (falsa) para obter a saída igual a 0.
Na disjunção é diferente, basta uma das condições seja igual a 1(verdadeira) para obter a saída igual a 1.

Exemplos de Atribuição de Dados

```
int a = 17, b = 3;
int x, y;
float z = 17. , z1, z2;
x = a / b;
y = a % b;
z1 = z / b;
z2 = a / b;
```

ao final da execução destas linhas, os valores calculados seriam
x = 5
y = 2
z1 = 5.666666
z2 = 5.0

Note que, na linha correspondente a z2, primeiramente é feita uma divisão inteira (pois os dois operandos são inteiros). Somente depois de efetuada a divisão é que o resultado é atribuído a uma variável float.

Os operadores de incremento e decremento são unários que alteram a variável sobre a qual estão aplicados. O que eles fazem é incrementar ou decrementar, a variável sobre a qual estão aplicados, de 1. Então

```
x++;
x--;
```

são equivalentes a

```
x=x+1;
x=x-1;
```

Expressões que Podem ser Abreviadas

O C admite as seguintes equivalências, que podem ser usadas para simplificar expressões ou para facilitar o entendimento de um programa.

Expressão Original	Expressão Equivalente
x = x + k;	x += k;
x = x – k;	x -= k;
x = x * k;	x *= k;
x = x/k;	x /= k;
x = x >> k;	x >>= k;
x = x << k;	x <<= k;
x = x & k;	x &= k;
etc...	

Sintaxe:

variável operador = expressão
//variável = variável operador expressão

Exemplo

x += 10; // é igual a x = x + 10;

y *= (10 + 20); // é igual a y = y *(10 + 20);

Encadeando expressões: o operador (,)
O operador vírgula (,) determina uma lista de expressões que devem ser executadas sequencialmente. Em síntese, a **vírgula** diz ao compilador: execute as duas expressões separadas pela vírgula, em sequência. O valor retornado por uma expressão com o operador (,) é sempre dado pela expressão mais à direita. No exemplo abaixo:

x = (y=2, y+3);

O valor 2 vai ser atribuído a y, se somará 3 a y e o retorno (5) será atribuído à variável x. Pode-se encadear quantos operadores (,) forem necessários.

O exemplo a seguir mostra outro uso para o operador (,) dentro de um for:

```
#include<stdio.h>
int main() {
        int x, y;
        for(x=0 , y=0 ;  x+y < 100 ; ++x , y++)

/* Duas variáveis de controle: x e y . Foi atribuído o valor zero a cada uma delas na inicialização do for e ambas são incrementadas na parte de incremento do for */

        printf("\n%d ", x+y);

/* o programa imprime os números pares de 2 a 98 */

}
```

Operador ?

O operador ternário (?) é utilizado para substituir uma instrução de desvio condicional como if - else. Ou seja, uma condição sendo testa.

Sua forma geral é:

(expressão1) ? expressão2 : expressão3;

Exemplo

```
resultado = (nota>7) ? 10 : 0;
```

Significa se a expressão for verdadeira executa a expressão 2, senão executa a expressão 3.

A variável resultado recebe (nota > 7), se essa condição for verdadeira então resultado recebe 10, caso contrário resultado recebe 0.

Operador sizeof

Este operador unário retorna o tamanho, em bytes, de uma determinada variável ou um tipo primitivo de dado.

Exemplo

```
        int a;
        char b;
        //Qual tamanho em bytes de uma variável tipo float?
```

```
a = sizeof(float);  //armazena o valor 4 em a

//Quantos bytes precisa para manipular a variável b ?
a = sizeof (b);  //armazena o valor 1 em a
```

Modeladores (Casts)

Este comando irá forçar uma expressão a ser de determinado tipo. Onde *tipo* é qualquer tipo de dado válido em C.
Sua forma geral é:

(tipo) expressão;

Exemplo

Dois números tipo inteiro, 5 e 2 e queremos dividir. Então vai retornar o resultado inteiro da divisão dos dois números (2.0). Para retornar, por exemplo, o resultado em valor real dessa divisão, ou seja, 2.5 colocamos antes entre parênteses float.

```
#include <stdio.h>
int main (){
        int num = 5;
        float f;
        f=(float)num/2;
/* Uso do modelador . Força a transformação de num em um float */
        printf ("%f",f);
        return(0);
}
```

Parênteses e Espaçamentos

Visando tornar expressões e o programa mais legível, devemos acrescentar espaçamentos e tabulações no código fonte. Assim como parênteses adicionais facilita a compreensão de uma expressão e dá mais controle ao programador referente às precedências de operações. Ou seja, importante usar endentações, parênteses e espaçamentos para melhor visibilidade do código fonte por parte do programador.