UNINOVE

Linguagem de Programação I

Prof. Marcos Alexandruk

alexandruk@uninove.br

HISTÓRICO

A Linguagem C foi projetada em 1972 no Laboratório da Bell por:

- Dennis M. Ritchie (em 1973, escreveu o UNIX utilizando C)
- Brian W. Kernighan

A linguagem C é derivada da ALGOL 68, baseada na linguagem B de Ken Tompson, uma evolução da BCPL.

O ANSI (American National Standards Institute) estabeleceu, em 1983, um comitê para criar um padrão para linguagem C.

VANTAGENS

- Poder de programação de médio nível
- Alto grau de portabilidade
- Apenas 32 palavras-chave (27 originais mais 5 adicionadas pelo ANSI)

PALAVRAS-CHAVE C 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
default	goto	sizeof	volatile
do	if	static	while

- Todas as palavras-chave da linguagem C são minúsculas

TIPOS DE DADOS

- **INTEIROS**

int	-32.768 a 32.767
unsigned int	0 a 65535
long	-2.147.483.648 a 2.147.483.647

- **REAIS**

float	3.4 E-38 a 3.4 E+38
double	1.7 E-308 a 1.7 E+308

- **CARACTERES**

char	0 a 255 caracteres

- **LÓGICOS (BOOLEANO)**

Na linguagem, não existe um tipo de dado lógico ou booleano.

Qualquer valor igual a 0 (zero) é considerado um valor lógico FALSO e qualquer valor diferente de zero (geralmente 1 inteiro) é considerado um valor lógico VERDADEIRO.

VARIÁVEIS

DECLARAÇÕES

Todo dado a ser armazenado numa variável deve ser previamente declarado: primeiro é necessário saber o seu tipo para depois fazer o seu armazenamento.

NOMES

- Poderão ser atribuídos com um ou mais caracteres;
- O primeiro caractere não poderá ser um número, deverá sempre ser uma letra;
- Não poderá conter espaços em branco;
- Não poderão ser utilizadas palavras reservadas da linguagem;
- Não poderão ser utilizados outros caracteres a não ser letras e números, exceto o _ (underscore);
- Apenas os primeiros 32 caracteres são significativos para maioria dos sistemas operacionais (para o MS-DOS apenas 8 caracteres);
- A linguagem *case sensitive* (há diferença entre caracteres maiúsculos e minúsculos).

CONSTANTES

DEFINIÇÃO

Tudo aquilo que é fixo ou estável (não apresenta alterações durante a execução do programa).

EXEMPLO

pi = 3.14159

OPERADORES ARITMÉTICOS

UNÁRIOS

Quando atuam na inversão de um valor, atribuindo a este o sinal positivo ou negativo.

BINÁRIOS

Quando atuam sobre dois valores: adição, subtração, multiplicação, divisão, exponenciação.

EXPRESSÕES ARITMÉTICAS

Seguem um formato diferente da forma conhecida em matemática.

Exemplo:

```
X = { 4 . [ 12 : ( 2 + 1 ) ] }
```

será escrita:

```
X = ( 4 * ( 12 / ( 2 + 1 ) ) )
```

ENTRADA, PROCESSAMENTO E SAÍDA DE DADOS

Uma entrada e uma saída de dados podem ocorrer de diversas formas:

ENTRADA: teclado, modem, leitores óticos, disco, etc.

SAÍDA: vídeo, impressora, disco, etc.

Considerando que as entradas serão feitas via teclado e as saídas via vídeo, os programas utilizarão as seguintes funções:

scanf() para ENTRADA de dados

printf() para SAÍDA de dados

ESTRUTURA DE UM PROGRAMA

Todo programa escrito em C consiste de uma ou mais **funções**, possibilitando a construção de programas modulares e estruturados.

A função **main()** é a principal instrução a ser considerada e deverá estar presente em algum lugar do programa. Ela marca o ponto de inicialização do processo de execução do programa.

```
<definições de pré-processamento>
<declaração de variáveis globais>
main()
{
   /*
      corpo da função, declaração de variáveis locais, etc.
   */
}
   <tipo> nome_da_funcao <parâmetros>
   <declaração de parâmetros>
   {
      /*
         corpo da função, declaração de variáveis locais, etc.
      */
   }
```

PRIMEIRO PROGRAMA EM C

ALGORITMO

1. Ler um valor para a variável A;
2. Ler um valor para a variável B;
3. Efetuar a adição da variável B, atribuindo o resultado à variável X;
4. Apresentar o valor da variável X.

PROGRAMA

```
main()
{
   int A;
   int B;
   int X;
   scanf("%d",&A);
   scanf("%d",&B);
   X = A + B;
   printf("%d",X);
}
```

scanf()

```
scanf("expressão_de_controle ", lista de argumentos);
```

A **expressão de controle** contém os códigos de formatação para o tipo de dado a ser processado, precedidos pelo sinal de percentagem %.

%c	Leitura de apenas um caractere
%d	Leitura de números inteiros decimais
%e	Leitura de números em notação científica
%f	Leitura de números reais (ponto flutuante)
%l	Leitura de número inteiro longo
%o	Leitura de um número octal
%s	Leitura de uma série de caracteres
%u	Leitura de um número inteiro decimal sem sinal
%x	Leitura de números hexadecimal

A **lista de argumentos** da função **scanf()** indica os endereços das variáveis em uso. O operador **&** possibilita retornar o conteúdo da variável.

Caso não seja usado o operador de endereço **&**, será retornado o endereço de memória em que se encontra a variável.

printf()

```
printf("expressão_de_controle ", lista de argumentos);
```

A **expressão de controle** contém os códigos de formatação para o tipo de dado a ser processado, precedidos pelo sinal de percentagem %.

%c	Escrita de apenas um caractere
%d	Escrita de números inteiros decimais
%e	Escrita de números em notação científica
%f	Escrita de números reais (ponto flutuante)
%l	Escrita de número inteiro longo
%o	Escrita de um número octal
%s	Escrita de uma série de caracteres
%u	Escrita de um número inteiro decimal sem sinal
%x	Escrita de números hexadecimal

SÍMBOLOS ESPECIAIS

\n	Gera uma nova linha
\t	Gera um espaço de tabulação
\b	Executa um retrocesso de espaço
\"	Apresenta o símbolo de aspas
\\	Apresenta o símbolo de barra invertida
\f	Gera um salto de página de formulário
\0	Gera um nulo

O uso dos símbolos especiais acima permite obter caracteres que não podem ser conseguidos diretamente do teclado.

EXEMPLO

Desenvolver um programa que efetue do cálculo do salário líquido de um profissional que trabalhe por hora. São dados: valor da hora de trabalho, número de horas trabalhadas no mês e percentual de desconto do INSS.

ALGORITMO

1. Ler a variável HT (horas trabalhadas)
2. Ler a variável VH (valor da hora de trabalho)
3. Ler a variável PD (percentual de desconto do INSS)
4. Calcular o salário bruto (SB): HT * VH
5. Calcular o total de desconto (TD): PD / 100
6. Calcular o salário líquido (SL): SB - TD
7. Apresentar os valores do salário bruto (SB), total de desconto (TD) e salário líquido (SL)

```
/* Calculo de Salario */
main()
{
   float HT,VH,PD,TD,SB,SL
   printf("Horas trabalhadas: ");
   scanf ("%f",&HT);
   printf("Valor da hora: ");
   scanf ("%f",&VH);
   printf("Percentual de desconto: ");
   scanf ("%f",&PD);
   SB=HT*VH;
   TD=(PD/100)*SB;
   SL=SB-TD;
   printf("Salario bruto ......: %f\n",SB);
   printf("Desconto ...........: %f\n",TD);
   printf("Salario liquido ....: %f\n",SL);
}
```

EXERCÍCIOS

1. Ler a temperatura em graus centígrados e apresentá-la convertida em graus Fahrenheit.
   Fórmula: F=(9*C+160)/5
   F= temperatura em Fahrenheit
   C= temperatura em Centígrados

2. Calcular e apresentar o valor do volume de um cilindro.
   Fórmula: pi x raio^2 x altura

3. Ler dois valores para as variáveis A e B, efetuar a troca dos valores de forma que a variável A passe a possuir o valor da variável B e que a variável B passe a ter o valor da variável A. Apresentar o valores trocados.

OPERADORES RELACIONAIS

==	IGUAL
!=	DIFERENTE
>	MAIOR
<	MENOR
>=	MAIOR OU IGUAL
<=	MENOR OU IGUAL

CUIDADO! Não confundir o operador de comparação **==** com o operador de atribuição **=**.

OPERADORES LÓGICOS

&& (AND)		
condição 1	condição 2	resultado
F	F	F
V	F	F
F	V	F
V	V	V

|| (OR)		
condição 1	condição 2	resultado
F	F	F
V	F	V
F	V	V
V	V	V

! (NOT)	
condição	resultado
F	V
V	F

ESTRUTURA DE DECISÃO: if

Se a condição for verdadeira será executada a instrução posicionada logo após a instrução if.

```
if (condição)
   instruções para condição verdadeira;
instruções seguintes;
```

Caso seja necessário incluir mais de uma instrução para condição verdadeira, estas deverão ser inseridas em blocos com os delimitadores { ... }.

```
if (condição)
   {
      instrução_1 para condição verdadeira;
      instrução_n para condição verdadeira;
   }
instruções seguintes;
```

EXEMPLO

Ler dois valores inteiros e independentemente da ordem em que foram entrados, estes deverão ser impressos na ordem crescente.

ALGORITMO

1. Atribuir o primeiro valor à variável A
2. Atribuir o segundo valor à variável B
3. Verificar se o valor de A é maior que o valor de B
   a. Se for verdadeiro, efetuar a troca de valores entre as variáveis
   b. Se for falso, executar o passo 4
4. Apresentar os valores das duas variáveis

```
/* ordena dois valores */
main()
{
   int A, B, X;
   printf("Informe um valor para A: ");
   scanf("%d", &A);
   printf("Informe um valor para B: ");
   scanf("%d", &B);
   if (A>B)
   {
      X=A;
      A=B;
      B=X;
   }
   printf("\nOs valores ordenados são: %d e %d", A, B);
}
```

ESTRUTURA DE DECISÃO: if ... else

Se a condição for verdadeira será executada a instrução posicionada logo após a instrução if, senão (se a condição for falsa) será executada a instrução posicionada logo após a instrução else.

```
if (condição)
   instruções para condição verdadeira;
else
   instruções para condição falsa;
```

Caso seja necessário incluir mais de uma instrução para condição verdadeira ou para condição falsa, estas deverão ser inseridas em blocos com os delimitadores { ... }.

```
if (condição)
   {
      instrução_1 para condição verdadeira;
      instrução_n para condição verdadeira;
   }
else
   {
      instrução_1 para condição falsa;
      instrução_n para condição falsa;
   }
```

EXEMPLO

Ler duas notas, somar seus valores e calcular a média (dividir o resultado da soma por dois). Se a média for maior ou igual a cinco, imprimir: APROVADO. Caso contrário, imprimir: REPROVADO.

ALGORITMO

1. Atribuir a primeira nota à variável A
2. Atribuir a segunda nota à variável B
3. Somar o valor de A ao valor de B e dividir por 2
4. Se o resultado for maior ou igual a cinco, imprimir: APROVADO
5. Se o resultado for menor que cinco, imprimir: REPROVADO

```
/* calcula média */
main()
{
   float A, B, X;
   printf("Informe a primeira nota: ");
   scanf("%d", &A);
   printf("Informe a segunda nota: ");
   scanf("%d", &B);
   X=(A+B)/2;
   if (X>=5)
   {
      printf("APROVADO");
   }
   else
   {
      printf("REPROVADO");
   }
}
```

EXERCÍCIOS

1. Ler dois valores numéricos e apresentar a diferença do maior para o menor.

2. Ler quatro valores referentes a quatro notas escolares de um aluno e imprimir uma mensagem dizendo que o aluno foi APROVADO se o valor da média for maior ou igual a 7. Se o valor da média for menor que 7, solicitar a nota de exame, somar com o valor da média e obter nova média. Se a nova média for maior ou igual a 5, apresentar uma mensagem dizendo que o aluno foi APROVADO EM EXAME. Caso contrário, informar que o aluno foi REPROVADO. Apresentar junto com as mensagens o valor da média do aluno para qualquer condição.

3. Ler três valores para os lados de um triângulo, considerando lados como: A, B e C. Verificar se os lados fornecidos formam realmente um triângulo: (A<B+C && B<C+A && C<A+B). Se a condição for verdadeira, indicar que tipo de triângulo foi formado: isósceles (dois lados iguais e um diferente), escaleno (todos os lados diferentes) ou eqüilátero (todos os lados iguais).

ESTRUTURA DE DECISÃO: switch

```
switch (variável)
{
   case constante_1:
      declaração_1
   break;
   case constante_2:
      declaração_2
   break;
   ...
   case constante_n:
      declaração_n
   break;
   default:
      declaração_default;
```

EXEMPLO:

```
main()
{
int A;
printf("Digite um número: ");
scanf("d%", &A);
switch (A)
{
   case 1:
      printf("\nO numero e igual a 1\n");
   break;
   case 1:
      printf("\nO numero e igual a 2\n");
   break;
   default:
      prinf("\nO numero e igual a 3\n");
}
```

OPERADOR TERNÁRIO: ?

```
main()
{
   int A=2, B;
   B = A<0 ? -1 : 1;
   printf("d%", B);
}
```

LOOPINGS

- Conhecidos também como laços ou malhas
- Utilizados para repetir um determinado trecho de um programa

WHILE

- Efetua um teste lógico no início de um looping
- Executa um conjunto de instruções enquanto a condição verificada for verdadeira

EXEMPLO

```
/* while */
main()
{
   int A, B, S, I;
   I = 1;
   while (I <= 3)
   {
      printf("\n\nInforme o valor de A: ");
      scanf("%d", &A);
      printf("Informe o valor de B: ");
      scanf("%d", &B);
      S = A + B;
      printf("\nResultado: %d", S);
      I++;
   }
}
```

DO ... WHILE

- Efetua um teste lógico no final de um looping
- Executa um conjunto de instruções pelo menos uma vez antes de verificar a validade da condição estabelecida

EXEMPLO

```
/* do ... while */
main()
{
   int A, B, S, I;
   I = 1;
   do
   {
      printf("\n\nInforme o valor de A: ");
      scanf("%d", &A);
      printf("Informe o valor de B: ");
      scanf("%d", &B);
      S = A + B;
      printf("\nResultado: %d", S);
      I++;
   }
      while (I <= 3)
}
```

FOR

Executa a inicialização incondicionalmente e testa a condição

EXEMPLO

```
/* for */
main()
{
   int I;
   for (I=1;I<=10;I++)
      printf("I = %d\n", I);
}
```

EXERCÍCIOS

1. Apresentar todos os valores numéricos inteiros ímpares situados na faixa de 1 a 20. Para verificar se o número é ímpar, efetuar dentro do looping a verificação lógica desta condição com a instrução if, perguntando se o número é ímpar; sendo, mostre-o, não sendo, passe para o próximo passo.

```
main()
{
   int n;
   for (n=1;n<=20;n++)
   {
      if ((n%2)==1)
      printf("%d\n",n);
   }
}
```

2. Apresentar o total da soma obtido dos cem primeiros números inteiros.

(1+2+3+...98+99+100)

```
main()
{
   int n, s=0;
   for (n=1;n<=100;n++)
   {
      s=s+n;
   }
      printf("%d",s);
}
```

3. Apresentar os resultados de uma tabuada para um número qualquer.

```
main()
{
   int n;
   for (n=1;n<=10;n++)
   {
      printf("%d x 3 = %d\n",n, n*3);
   }
}
```

4.Escreva um programa que apresente a série de Fibonacci até o décimo termo. A série de Fibonacci é formada pela seqüência: 1, 1, 2, 3, 5, 8, 13, 21 ...

```
main()
{
   int a=1, b=1, i;
   for (i=1;i<=5;i++)
   {
      printf("%d %d ",a, b);
      a=a+b;
      b=a+b;
   }
}
```

VETORES (MATRIZES DE UMA DIMENSÃO)

- Utilizados quando tipos básicos (inteiro, real, caractere e lógico) e/ou variáveis simples não são suficientes para representar a estrutura de dados utilizada em um programa.
- A dimensão de uma matriz é constituída por constantes inteiras e positivas.
- Os nomes dados às matrizes seguem as mesmas regras de nomes utilizados em variáveis simples.
- Um vetor ou matriz de uma dimensão é representado por seu nome, tamanho (dimensão) entre colchetes e seu tipo.

SINTAXE

```
tipo nome_da_matriz[dimensão]
```

onde:

- **tipo** tipo de dado a ser guardado na matriz
- **dimensão** tamanho da matriz em número de elementos

EXEMPLO

```
/* calculo da media de 10 alunos */
main()
{
   float M[10];
   float MEDIA, SOMA;
   int I;
   char TECLA;
      printf("\nCALCULO DA MEDIA\n\n");
      for (I=0;I<10;I++)
      {
         printf("Informe a %da. Nota: " I+1);
         scanf("%f", &M[I]);
         SOMA += M[I];
      }
      MEDIA = SOMA/10;
      printf("\nMedia dos 10 alunos: %6.2f\n", MEDIA);
      printf("\nTecle algo para encerrar: ");
      TECLA = getche();
}
```

EXERCÍCIO

1. Efetuar a leitura de 5 elementos (números inteiros) de uma matriz M do tipo vetor. No final, apresentar o total da soma de todos os elementos ímpares.

```
/* calculo dos elementos impares de um vetor */
main()
{
   int M[5];
   int I, SOMA=0;
   char TECLA;
      printf("\nSOMA DE ELEMENTOS IMPARES\n\n");
      for (I=0;I<5;I++)
      {
         printf("Informe o valor do %do. elemento: " I+1);
         scanf("%d", &M[I]);
      }
      for (I=0;I<5;I++)
         if (M[I]%2==1)
         SOMA += M[I];
      }
      printf("\nSoma dos elementos impares: %d", SOMA);
      printf("\nTecle algo para encerrar: ");
      TECLA = getche();
}
```

STRINGS

- Utilizadas para manipulação e armazenamento de textos.
- Caracteriza-se por ser um vetor (matriz de uma dimensão) do tipo **char** terminado com o caractere NULL "\0".
- O caractere NULL não é visível, mas é muito importante: é a única forma que as funções possuem para identificar o final de uma string.
- As strings variáveis, fornecidas via teclado, são tratadas através da função **scanf()** com a utilização do formato **%s**.

EXEMPLO

```
main()
{
int NOME[10];
printf()("Digite o seu nome:");
scanf("%s", NOME);
printf()("Ola %s, NOME");
}
```

- Se a matriz possui declarado um valor de tamanho 10, você só poderá utilizar 9, deixando uma posição livre para o caractere "\0".
- Observe a ausência do operador **&**, precedendo o segundo argumento das instruções de entrada da função **scanf()**. Este operador não deve ser usado, pois o nome de uma matriz é o seu endereço inicial.