**Linguagem C: Comandos Básicos**

Um programa em C é um conjunto de funções definidas pelo programador, funções que utilizarão outras funções definidas pelo programador e algumas funções oferecidas pelo sistema (as funções oferecidas pelo sistema sao chamadas *funções de biblioteca* ou funções predefinidas).

**Estrutura de um programa em C**

main()

{

}

**Tipos de Dados**

|  |  |  |
| --- | --- | --- |
| **Tipo** | **Número de bytes** | **Faixa de valores** |
| char | 1 | Caracteres da tabela ASCII |
| int | 2 | Números inteiros de –32768 a 32767 |
| long ou long int | 4 | Números inteiros de –65536 a 65535 |
| float | 4 | Números reais de –3,4x1038 a –3,4x10-38 e 3,4x10-38 a 3,4x1038 |
| double | 8 | Números reais de –1,7x10308 a -1,7x10-308 e 1,7x10-308 a 1,7x10308 |
| void | 0 | Conjunto vazio |

**Variáveis**

<tipo> <identificador>;

<tipo> <identificador> = <valor-inicial>;

<tipo> <lista-ident-sep-por-vírgula>;

**Exemplos:**

char caractere;

int idade;

long totalAlunos;

float valor;

double dividaExterna;

**Comentários**

Usando // comenta-se apenas a linha corrente a partir de onde o // foi colocado. Já usando /\* \*/, o texto que fica entre /\* e \*/ torna-se um comentário, mesmo que esteja contido em mais de uma linha.

main()

{

// Isto é um comentário

/\* Isto também é um comentário

Continua sendo comentário aqui \*/

}

**Constantes**

const <tipo> <identificador> = <valor>;

**Exemplos:**

const float PI = 3.1416;

const int MIN = 0, MAX = 100;

**Comandos de atribuição**

<var> = <valor>;  
<var>++;  
<var>--;  
++<var>;  
--<var>;

**Exemplos:**

x = 1;

x++; // incrementa o valor de x para 2

x--; // decrementa o valor de x para 1

y = x++; // y recebe o valor de x (1) e x é incrementado para 2

z = ++x; // o valor de x é incrementado para 3 e depois z recebe 3

O comando de atribuição, além de guardar o valor na variável, retorna este valor, o que permite que um comando de atribuição seja usado como uma expressão (em um if, por exemplo).

**Expressões aritméticas**

|  |  |
| --- | --- |
| **Operador** | **Operação** |
| + | adição |
| - | subtração |
| \* | multiplicação |
| / | divisão |
| % | módulo (resto da divisão) |

**Exemplos:**

2 + 2 // resulta em 4;

10 % 3 // resto da divisão de 10 por 3 = 1

**Associando Atribuições e Operações Aritméticas**

|  |  |
| --- | --- |
| **Operador** | **Operação** |
| += | adição e atribuição |
| -= | subtração e atribuição |
| \*= | multiplicação e atribuição |
| /= | divisão e atribuição |
| %= | módulo e atribuição |

**Exemplos:**

int x = 0;

x += 3; // equivale a x = x + 3;

x \*= 2; // equivale a x = x \* 2;

x -= 1; // equivale a x = x - 1;

**Operadores Relacionais**

Quando a operação relacional resulta em verdadeiro, ela retorna o valor 1. Quando falso, retorna o valor 0.

|  |  |
| --- | --- |
| **Operador** | **Operação** |
| > | maior |
| < | menor |
| >= | maior ou igual |
| <= | menor ou igual |
| == | igual |
| != | diferente |

**Exemplos:**

2 == 2 // 2 é igual a 2?

3 != 2 // 3 é diferente de 2?

3 >= 2 // 3 é maior ou igual a 2?

**Expressões Lógicas**

|  |  |  |
| --- | --- | --- |
| **Operador** | **Operação lógica** | **Tipo** |
| && | e | Binário |
| || | ou | Binário |
| ^^ | ou exclusivo | Binário |
| ! | negação | Unário |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Avaliação de Expressões Lógicas** | | | | | | |
| ***A*** | ***B*** | ***A && B*** | ***A || B*** | ***A ^^ B*** | ***!A*** | ***!B*** |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**Entrada dos dados**

A função *scanf* é usada para efetuar a entrada de dados. Ela está disponível na biblioteca de funções stdio (*standard input output*). Para que seja possível usar a função *scanf*, é preciso fazer um *include* (inclusão) do arquivo de cabeçalho *stdio.h* no programa.

A função *scanf* recebe uma *expressão de controle* seguida de uma *lista de argumentos*. A expressão de controle contém mensagens e *códigos de formatação* (veja tabela abaixo) que indicam o tipo de dado esperado na expressão de controle. Os valores lidos são armazenados em seqüência na lista de argumentos (variáveis) passada após a expressão de controle. É importante notar que os nomes das variáveis devem ser precedidos de & para indicar que será passada a referência para a variável e não uma cópia do valor da mesma. Isso permite que a função altere internamente o valor do parâmetro associado à variável e esta alteração seja refletida na variável passada.

|  |  |
| --- | --- |
| **Tabela de Conversão da função** ***scanf*** | |
| **Código de formatação** | **Valor Armazenado** |
| %c | um caractere |
| %d | um inteiro no sistema decimal |
| %i | um inteiro no sistema decimal |
| %o | um inteiro no sistema octal |
| %x | um inteiro no sistema hexadecimal |
| %ld | um inteiro do tipo long |
| %e | um número na notação científica |
| %f | um número em ponto flutuante |
| %s | uma cadeia de caracteres |

**Exemplo:**

// leitura de um caractere

char letra;

scanf("%c", &letra);

// Leitura de dia mes e ano

int dia, mes, ano;

scanf("%d/%d/%d", &dia, &mes, &ano);

Para a leitura de um caractere, podemos usar as funções *getch* e *getche*, que capturam e retornam o caractere digitado no teclado. A diferença entre as duas funções é que *getch* não exibe o caractere digitado na tela. Ambos são definidos na biblioteca de funções *conio*.

**Exemplo:** Lendo caracteres e repetindo 3 vezes

main() {

char c = getch();

printf("Caractere digitado: %c%c%c", c, c, c);

printf("\n");

char c2 = getche();

printf(" Caractere digitado: %c%c%c", c2, c2, c2);

}

**Saída de dados**

A saída de dados pode ser feita através da chamada à função *printf* também pertencente à biblioteca de funções *stdio*. Desta forma, o procedimento de declaração *include* desta biblioteca é também requerido.

**Exemplo:** Usando códigos de formatação

main() {

printf("%c", 'A'); // exibe o caractere A

printf("%d", 'A'); // exibe o número 65

printf("%d", 65); // exibe o número 65

printf("%c", 65); // exibe o caractere A

printf("%o", 65); // exibe o número octal 101

printf("%x", 65); // exibe o número hexadecimal 41

}

**Exemplo:** Leitura das partes de uma data

main() {

// Leitura de dia mes e ano

int dia, mes, ano;

printf("Digite uma data: ");

scanf("%d/%d/%d", &dia, &mes, &ano);

// Imprime dia, mes e ano separadamente

printf("dia %d\n", dia);

printf("mes %d\n", mes);

printf("ano %d\n", ano);

}

**Exemplo:** Formatação de números fracionários

main() {

printf("%8.2f\n", 12.54321);

printf("%8.2f\n", 123.54321);

printf("%8.2f\n", 1234.54321);

printf("%8.2f\n", 12345.54321);

}

O resultado da execução do programa acima é mostrado abaixo. O código de formatação %8.2f indica que o formato de impressão do número do tipo float será de tamanho total 8, sendo 2 o tamanho da parte fracionária.

12.54

123.54

1234.54

12345.54

**Códigos Especiais**

De acordo com a tabela de códigos especiais abaixo, podemos exibir alguns caracteres, como ", % e \, dentro de uma mensagem e a realização de ações que permitem a formatação da saída, como quebra de linha, tabulação e retrocesso.

|  |  |
| --- | --- |
| **Tabela de Códigos Especiais** | |
| **Código** | **Ação** |
| \n | executa uma quebra de linha |
| \t | executa uma tabulação |
| \b | executa um retrocesso (*backspace*) |
| \f | leva o cursor para a próxima página |
| \a | executa um sinal sonoro (beep) |
| \" | mostra o caractere " |
| \\ | mostra o caractere \ |
| \% | mostra o caractere % |

**Exemplos:**

main() {

printf("abc\b\b\bABC\n"); // Imprime abc, apaga e depois imprime ABC e quebra a linha

printf("xyz\tXYZ"); // Imprime xyz, a tabulação e XYZ

printf("\\"); // Imprime uma barra

printf("\""); // Imprime aspas

}

**Funções Predefinidas**

A tabela abaixo contém uma lista das funções predefinidas disponíveis.

|  |  |  |
| --- | --- | --- |
| **Função** | **Parâmetros** | **Retorna** |
| tolower(c) | char | caractere minúsculo correspondente a c |
| toupper(c) | char | caractere maiúsculo correspondente a c |
| fabs(n) | double | valor absoluto de n |
| sqrt(n) | double | raiz quadrada de n |
| power(x,y) | double, double | x elevado a y |
| sin(x) | double | seno de x |
| cos(x) | double | coseno de x |
| tan(x) | double | tangente de x |
| log(n) | double | logaritmo natural de n |
| log10(n) | double | logaritmo decimal de n |