Universidade de São Paulo - São Carlos,SP
Instituto de Ciências Matemáticas e de Computação

USP - ICMC - SSC
SSC 0501 - 1o. Semestre 2011

Disciplina de
Introdução à Ciência da Computação
ICC 1 - Teoria

Prof. Fernando Santos Osório
Email: fosorio [at] { icmc. usp. br , gmail. com }
Página Pessoal: http://www.icmc.usp.br/~fosorio/
Web - WIKI ICMC: http://wiki.icmc.usp.br/index.php/SSC-501

PAE: Daniel Sales (Mestr. CCMC – LRM)
Email: dsales [at] icmc.usp.br

Aula 02t

Aula 02 – Algoritmos e Programação em "C"

Agenda:

1. Linguagem "C" - Conceitos
   - Estrutura de um programa em "C"
   - Tipos de Dados
   - Declaração e Uso de Variáveis
   - Operadores Aritméticos
   - Expressões Aritméticas
   - Entrada e Saída (E/S) básica da Linguagem "C":
     Comandos - printf e scanf
2. Prática de programação – Programas Sequenciais

Algoritmos => Programas

Programa de Computador: Comandos

Comandos são ordens para que o computador manipule os dados de sua memória...

Exemplo de Programa: LINGUAGEM "C"

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    printf("Hello World\n");
    system("PAUSE");
    return 0;
}
```


Algoritmos => Programas

Programa de Computador: Comandos

```
/* Comentário: Este é um Exemplo de Programa em "C" */

#include <stdio.h>
#include <stdlib.h>
char Nome[30];

int main(int argc, char *argv[])
{
    printf("Qual o seu nome? ");
    scanf ("%s",Nome);
    printf("Hello %s\n",Nome);
    system("PAUSE");
    return 0;
}
```

Linguagem "C"

Programação em C

VARIÁVEIS E TIPOS DE DADOS

Variáveis

- Nomes de Variáveis:

  - Variáveis recebem nomes (etiquetas)
  - Nomes com Letras, Numeros e '_'
  - Nomes começam por Letra ou '_'
  - Não deve conter caracteres "estranhos"
  - Não pode usar palavras reservadas
  - Minúsculas e Maiúsculas podem ser usadas onde são diferenciadas: Nome é ≠ de NOME

Tipos de Dados

Modificador	Tipo	Descrição
	CHAR	Caracter ASCII (valores de 0 a 255 / -128 a + 127 = 1 byte)
UNSIGNED	CHAR	Byte com valores positivos, sem sinal (0 a 255)
SIGNED	CHAR	Byte com valores negativos, com sinal (-128 a +127)
	CHAR[x]	String de no máximo 'x'-1 componentes (caracteres)
	INT	Valor inteiro (usualmente de 2 ou de 4 bytes)
UNSIGNED	INT	Valor inteiro positivo
SIGNED	INT	Valor inteiro positivo ou negativo, com sinal
SHORT	INT	Valor inteiro com precisão inferior
LONG	INT	Valor inteiro com precisão superior
	INT[x]	Vetor de inteiros com 'x' componentes (valores inteir
	FLOAT	Valor de ponto flutuante (c/casas decimais)
UNSIGNED	FLOAT	Valor de ponto flutuante positivo
	DOUBLE	Valor de ponto flutuante com dupla precisão
LONG	DOUBLE	Valor de ponto flutuante com dupla precisão estendida
	ENUM	Enumeração de elementos. Exemplo: Seg, Ter, Qua, Qui, S
	VOID	Tipo de dados indefinido. Indica apenas o endereço da memória, sem no entanto especificar o tipo exato.

Modificadores: Signed, Unsigned, Short, Long, Static, Register

Tipos de dados básicos em C

Tipos mais comuns

- char: um byte que armazena o código de um caractere do conjunto de caracteres local (ASCII)
- int: um inteiro cujo tamanho depende do processador e do compilador usado, tipicamente 16 ou 32 bits (2 ou 4 bytes)
- float: um número real com precisão simples
- double: um número real com precisão dupla

Como saber o tamanho/precisão de um tipo de dado?

#include <limits.h> => Arquivo: C:\Dev-Cpp\include\limits.h

Declaração de variáveis

```
/* Declarando valores constantes... */

#define  MAXIMO   1000          /* Início: Isto é um comentário...          Fim */
#define  VERSAO   1.0           /* O #define é uma macro do pré-processador */
#define  SISOP    "LINUX"
const    double   valor_do_pi = 3.1415926;   /* Cria uma "variável constante"   */

/* Declarando variáveis... */

int idade;                  /* Variáveis podem ser declaradas antes de começar */
double salario;             /*  a parte que descreve as rotinas do programa    */
char sexo;                  /* Estas variáveis são chamadas de GLOBAIS         */

int main( )
{                           /* Sempre após um "abre chaves" posso declarar vars. */
  int dia;                  /* Estas variáveis pertencem ao bloco definido entre o */
  int mes;                  /* abre chaves '{' e o fecha chaves '}'                */
  int ano;                  /* Estas variáveis são chamadas de LOCAIS do bloco     */
  ...
}
```

Observações importantes: VARIÁVEIS DEVEM SER DECLARADAS PARA SEREM USADAS!
- Nomes de variáveis devem começar por uma letra, seguidos de letras, dígitos ou '_';
- Maiúsculas e minúsculas são diferenciadas no nome das variáveis;
- Atenção para não declarar variáveis com nomes de palavras reservadas: for, while, int, ...

Usando as variáveis

- Atribuindo valores na declaração:

```
float  cotacao_dolar = 1.87;
char  ativo = 'S';
```

- Atribuindo valores diretamente nas variáveis:

```
int  dia;                 dia = 21;
float salario;            salario = 100.00;
double tabela[10];        tabela[0] = 0.001;
char nome[30];            strcpy(nome,"Fulano");
```

- Expressões:

```
int a,b;  double c;       c = a / b;      /*  Cuidado com os tipos de dados!! */
                          c = a + b / 2.0;  /* Cuidado com a precedência!! */
```

- Operador de conversão explícita de tipos de dados: "cast"

```
int valint;               valdoub = (double)valint;
double valdoub;           valint    = (int)valdoub;     /* Conversão explícita    */
                          valint    = valdoub;          /* Conversão implícita    */
                          valdoub  = valint/(double)outro_valint;  /* Forçada  */
```

Linguagem "C"

Programação em C

OPERADORES

Operadores

* OPERADOR DE ATRIBUIÇÃO :

= Atribui um valor ou resultado de uma expressão contida a sua direita para a variável especificada a sua esquerda.
Exemplo: A = 10; B = C * VALOR + GETVAL(X);
A = B = C = 1; /* Aceita associação sucessiva de valores */

* OPERADORES ARITMÉTICOS :

Operam sobre números e expressões, resultando valores numéricos

+ soma (int, float, double)
- subtração (int, float, double)
* multiplicação (int, float, double)
/ divisão (int, float, double)
% módulo da divisão (resto da divisão inteira)
- sinal negativo (operador unário)

* OPERADORES RELACIONAIS (Igualdade):

== Testa se são iguais. Exemplo: IF (a == b)
!= Testa se são diferentes. Exemplo: IF (a != b)

Operadores

• OPERADORES RELACIONAIS :

Operam sobre números e expressões, resultando valores lógicos (verdadeiro/falso)
Importante: Em "C" o valor **falso** vale 0 (**zero**) e
o valor **verdadeiro** é !0 (**diferente de zero**)

==	Testa se são iguais. Exemplo:	IF (a == b)
!=	Testa se são diferentes. Exemplo:	IF (a != b)
>	Maior que... Exemplo:	IF (a > b)
<	Menor que... Exemplo:	IF (a < b)
>=	Maior ou igual a... Exemplo:	IF (a >= b)
<=	Menor ou igual a... Exemplo:	IF (a <= b)

• OPERADORES LÓGICOS:

Operam sobre números e expressões, resultando valores lógicos (verdadeiro/falso).
Importante: 0 é Falso; 1 ou valores diferentes de 0 é Verdadeiro

! Negação (**not**). Inverte o valor lógico. Exemplo: a = !b;
&& Conjunção "E" (**and**). Ambos tem que ser verdadeiro. Exemplo: IF (a && b)
|| Disjunção "OU" (**or**). Qualquer um dos dois pode ser verdadeiro.
Exemplo: IF (a || b)

Operadores

Atenção:

- Expressões aritméticas seguem a precedência usual dos operadores aritméticos: * , / tem precedência sobre + , -

- Use (e abuse) sempre dos parênteses.

- Expressões aritméticas possuem regras próprias de conversão do tipo do dado usado na expressão!

- Use (e abuse) sempre da conversão explícita de tipos ("cast").
Exemplos:

Media = Nota1 + Nota2 + Nota3 / 3; /* Errado! */

Media = (double)(Nota1 + Nota2 + Nota3) / 3.0; /* Ideal */

"cast"

Operadores Aritméticos

- Representam as operações aritméticas básicas

Operação	Operador
Adição	+
Subtração	-
Multiplicação	*
Divisão	/
Resto da Divisão	%
Incremento (+1)	++
Decremento (-1)	--
Sinal Negativo	-

Operadores Relacionais

- Estabelecem relações/comparações

Operação	Operador
Igualdade	==
Diferença	!=
Maior	>
Maior ou igual	>=
Menor	<
Menor ou igual	<=

Operadores Lógicos

- Representam as operações básica dada na lógica matemática

Operação	Operador
Negação	!
Conjunção (E)	&&
Disjunção (OU)	||

Linguagem "C"

Programação em C

ESTRUTURA BÁSICA

Programação em C

- Todo programa, escrito na linguagem C, deve apresentar uma função principal chamada main, que define todo o corpo do programa
- Exemplo:

```
int main()
{
   /* corpo do programa */
}
```


Programação em C

- Todo programa, escrito na linguagem C, deve apresentar uma função principal chamada main, que define todo o corpo do programa
- Exemplo: Um pouco mais completo...

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char *argv[])
{
    /* corpo do programa */
    system ("PAUSE");
    return 0;
}
```

Comandos de Saída

- Empregados para que o sistema forneça, em um dispositivo de saída, as mensagens e resultados de seu processamento.
- O dispositivo padrão de saída é o monitor.
- A linguagem C oferece alguns comandos de saída, mas o que apresenta propósito mais geral é o printf.

Comando PRINTF()

- Sintaxe:

  printf("Mensagem", lista de variáveis);

- Funcionamento:
  - O comando escreve a mensagem dada no dispositivo padrão de saída, realizando a substituição das máscaras de formatação encontradas pelas respectivas variáveis dadas na lista subseqüente a mensagem.
  - O dispositivo padrão é dado pela variável stdout

Máscaras de formatação

- Símbolo de por cento seguido de uma letra:

%c Caractere
%d Inteiros com sinal
%u Inteiros sem sinal
%f Números reais (float)
%lf Números reais (double ou long float)
%s Cadeia de caracteres (strings)
%e Notação científica
%x Números em hexadecimal

Exemplo

- **PRINTF - Escrita de dados na tela (stdout)**

printf (<string_de_controle> , <variável>, <variável>, ...);

Exemplos:

```
printf ("Escrevendo apenas um texto na tela.");
printf ("No final pula para a linha seguinte.\n");
printf ("Valor inteiro = %d",variavel_int);
printf ("Valor float    = %f",variavel_float);
printf ("Valor double = %lf",variavel_double);
printf ("Caracter       = %c",variavel_letra);
printf ("Palavra (sequencia de caracteres) = %s",nome);
printf ("Números: %d, %lf, %d\n", 2, 3.1415, variavel_int);
printf ("Valor só com 3 casas após a vírgula: %.3lf",variavel_double);
printf ("Hoje é %d/%d/%d\n",dia, mes, ano);
```

Constantes do Tipo Char

- Barra invertida seguido de um caractere:

\a	bip
\b	backspace
\n	nova linha
\t	tabulação horizontal
\'	apóstrofe
\"	aspas
\\	barra invertida
\f	form feed

Comandos de entrada

- Utilizado para receber dados fornecidos pelo usuário (dados de entrada) e armazená-los na memória principal (em variáveis)
- Os dados são fornecidos ao sistema por meio de um dispositivo de entrada, cuja configuração dada como padrão é o teclado.
- A linguagem C oferece vários comandos de entrada, cada qual mais indicado para uma situação em particular.
- O principal comando de entrada é o scanf

Comando SCANF()

- Sintaxe:

  scanf("formato", &variável);

- Funcionamento:
  - O comando coleta as informações dadas no dispositivo padrão de entrada, interpretando as informações segundo a máscara de formatação e armazenando na(s) respectiva(s) variável(is) dada(s) subseqüentemente ao formato.
  - O dispositivo padrão é dado pela variável stdin

Exemplo

- Entrada formatada scanf().
  - Exemplos:

```
int idade; float salario; double x; char nome[10];
scanf("%d",&idade);
scanf("%f",&salario);
scanf("%lf",&x);
scanf("%s",nome);
```

  - Ou ainda:

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

Exemplo

- Entrada formatada scanf().
  - Exemplos:

```
int idade; float salario; double x; char nome[10];
scanf("%d",&idade);
scanf("%f",&salario);
scanf("%lf",&x);
scanf("%s",nome);
```

  - Ou ainda:

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


Exemplo

•**SCANF - Leitura de dados do teclado (stdin)**

scanf (<string_de_controle> , &<variável>, &<variável>, ...);

Exemplos:

```
int variavel_int;          double variavel_double;
float variavel_float;      char variavel_char;
int vint, dia, mes, ano;   char nome[30];

scanf ("%d", &variavel_int);
scanf ("%f", &variavel_float);
scanf ("%lf", &variavel_double);
scanf ("%c", &variavel_char);      /* Humm... Pode não funcionar bem */
scanf ("%s", nome);                /* String: NÃO tem o '&' !!! */
scanf ("%d %lf %d", &vint, &vdouble, &vint);
scanf ("%d %d %d", &dia, &mes, &ano);
```

Linguagem "C"

Programação em C

UM EXEMPLO COMPLETO

Programa C

```
/* Programa: calculo da área e do perímetro
             de uma circunferência          */

#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Programa C

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Informa ao pré-processador os arquivos de funções (bibliotecas) que precisam ser vinculados a esse programa para a construção do arquivo executável

Programa C

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Função Principal: main()
Rotina fundamental de todo o programa C,
É o “início” do algoritmo

Programa C

Bloco de Comandos: { ... }
Define as ação/instruções que serão executadas;
Tem seu início e fim delimitado por chaves

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```


Programa C

raio, **area** e **perim** são nomes identificadores de variáveis e precisam ser declarados antes de serem utilizados.

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Programa C

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Comando de saída: printf (...)
Escreve a mensagem
"Forneça o valor do RAIO:"
na tela

Programa C

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Comando de entrada: scanf (...)
Realiza a leitura do teclado, interpretando a informação como um número inteiro e armazenando na variável raio

Programa C

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Operador de Atribuição: **=**
Armazena o resultado da expressão na variável dada a esquerda, isto é, na variável area

Programa C

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Operador de Atribuição:
Armazena o resultado da expressão na variável perim

Programa C

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Constantes: M_PI
Dadas por identificadores, representam valores específicos. Neste caso, equivale ao valor de π e está definida na biblioteca MATH.H

Programa C

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Comando de saída: printf (...)
Escreve na tela a mensagem e o valor armazenado na variável area, saltando para a próxima linha ao término

Programa C

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Comando de saída: printf(...)
Escreve na tela a mensagem e o valor armazenado na variável perim, saltando para a próxima linha ao término

Programa C

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Comando de saída: printf(...)
Salta para a próxima linha e escreve a mensagem "Pressione qq tecla para continuar ..."

Programa C

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Comando de entrada: getch()
Aguarda até que o usuário pressione alguma tecla, retornando seu valor para o programa (que para nesse caso não está sendo armazenado na memória)

Programa C

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    printf("\nPressione qq tecla para retornar ...");
    getch();
    return 0;
}
```

Retorna o valor 0 ao Sistema Operacional, como um código indicando sucesso na execução do programa

Programa C

```
#include <stdio.h>
#include <math.h>

int main()
{
    int raio;
    float area, perim;

    printf("Forneca o valor do RAIO: ");
    scanf("%d",&raio);
    area = M_PI * raio * raio;
    perim = 2 * M_PI * raio;
    printf("Area: %f\n",area);
    printf("Perimetro: %f\n",perim);
    system("pause");

    return 0;
}
```


Programação em C

EXERCÍCIOS:

1) Faça um programa que leia duas notas de um aluno (nota 1 e nota 2) fornecidas pelo usuário que irá entrar as notas digitando pelo teclado. Usando estas 2 notas, calcule a média simples do aluno, e depois mostre na tela o resultado da média calculada.

2) Faça um programa que leia 3 notas de um aluno, onde a primeira e segunda nota possuem peso 1 e a terceira nota possui peso 2. Calcule a média ponderada destas notas, usando os pesos, e depois mostre na tela o resultado (exibir a média com apenas 2 casas após a vírgula).

3) Faça um programa que leia uma temperatura fornecida em graus Celsius (°C) e converta para graus Fahrenheit (°F), exibindo o resultado na tela.

4) Faça um programa que leia o valor da hora de trabalho (em reais) e número de horas trabalhadas no mês, e exiba na tela o valor a ser pago ao funcionário, adicionando 10% sobre o valor calculado.

INFORMAÇÕES SOBRE A DISCIPLINA

USP - Universidade de São Paulo - São Carlos, SP
ICMC - Instituto de Ciências Matemáticas e de Computação
SSC - Departamento de Sistemas de Computação

Prof. Fernando Santos OSÓRIO
Web institucional: http://www.icmc.usp.br/ssc/
Página pessoal: http://www.icmc.usp.br/~fosorio/
E-mail: fosorio [at] icmc. usp. br ou fosorio [at] gmail. com
PAE Gustavo Pessin – E-mail: pessin [at] gmail .com

Disciplina de Introdução a Computação – Eng. Ambiental
Web disciplina: COTEIA - Http://coteia.icmc.usp.br
> Programa, Material de Aulas, Critérios de Avaliação,
> Trabalhos Práticos, Datas das Provas, Notas

Material Complementar...

CONVERSÃO DE TIPOS :

O "C" segue algumas regras para a conversão de tipos de dados para possibilitar ao compilador de realizar operações em expressões com tipos compatíveis entre si.

As regras de conversão de tipos são as seguintes :

- Todos os valores não-inteiros ou não-double são convertidos como é mostrado na tabela indicada ao lado. Após isto os dois valores a serem operados serão ou do tipo int (incluindo long unsigned) ou do tipo double.
- Se um dos operandos é do **tipo double**, o outro operando também será convertido **para double**.
- Por outro lado, se um dos operandos for do **tipo unsigned long**, o outro será convertido **para unsigned long**.
- Por outro lado, se um dos operandos for do **tipo long**, então o outro operando será convertido **para long**.
- Por outro lado, se um dos operandos for do **tipo unsigned**, então o outro operando será convertido **para unsigned**.
- E por último, **ambos os operandos serão do tipo int.**

Material Complementar...

CONVERSÃO DE TIPOS :

O "C" segue algumas regras para a conversão de tipos de dados para possibilitar ao compilador de realizar operações em expressões com tipos compatíveis entre si.

TIPO	CONVERTIDO PARA	MÉTODO USADO
char	int	extensão de sinal
unsigned char	int	zera o byte mais significativo
signed char	int	extensão de sinal
short	int	se unsigned, entao unsigned int
enum	int	mesmo valor
float	double	preenche mantissa com 0's

Resumindo: "prioridade"
double, unsigned long, long, unsigned, int

Exemplo :

```
char  ch;   result = ( ch / i )   +   ( f * d )   -   ( f + i )
int    i;              |   |          |   |          |   |
float  f;             int  |        double |        double |
double d;                \  /           \  /           \  /
                          int          double          double
double result;             +----double----+               |
                                     +--------double--------+
```


Material Complementar...

Exemplos de constantes : Pré-definidas

DEFINE <nome_const> <valor>

```
255     -> Decimal      * constantes Inteiras
0777    -> Octal        * começando com 0 -> Octal
0XFF    -> Hexadecimal  * começando com 0x -> Hexa
nnnL    -> Long         * L,U -> forçam uso do tipo
nnnU    -> Unsigned     * podem ser usados juntos
nnnF    -> Float        * força armazenamento como float
'c'     -> Caracter (seu valor é o ascii de 'c' - tipo char)
'CC'    -> Caracter ( 2 caracteres - tipo int ) (TC)
           \XXX   -> Octal XX (expl.: '\014' é o form feed).
                     São esperados 3 dígitos após o "\" !
\n      -> New Line       LF  0x0A   * Sequências de escape
\b      -> Backspace      BS  0x08
\r      -> Carriage Return  CR  0x0D
\f      -> Form Feed      FF  0x0C
\t      -> Tab Horiz      HT  0x09
\v      -> Tab Vertical   VT  0x0B
\0      -> Null           0x00
\\      -> Backslash      \   0x5C
\'      -> Single quote   '   0x2C
\"      -> Double quote   "   0x22
\?      -> Question Mark  ?   0x3F
\a      -> Bell           BEL 0x07  (ANSI)
\xNN    -> Repr. hexa     0xNN  (ANSI)
```

Material de Aula

- Material disponível na CoTeia
  - Transparências de Aula
  - Lista de Exercícios
  - Exemplos de Programas
- Agradecimentos

Fonte do material cedido que foi usado na preparação desta aula:
Algoritmos - Leandro Fernandes
Introdução Programação C – Leandro Fernandes

Universidade de São Paulo - São Carlos,SP
Instituto de Ciências Matemáticas e de Computação

INFORMAÇÕES SOBRE A DISCIPLINA

USP - Universidade de São Paulo - São Carlos, SP
ICMC - Instituto de Ciências Matemáticas e de Computação
SSC - Departamento de Sistemas de Computação

Prof. Fernando Santos OSÓRIO
Web institucional: http://www.icmc.usp.br/ssc/
Página pessoal: http://www.icmc.usp.br/~fosorio/
E-mail: fosorio [at] icmc. usp. br ou fosorio [at] gmail. com
PAE Daniel Sales – E-mail: dsales [at] icmc.usp.br

Disciplina de Introdução a Ciência da Computação
Web disciplina: Wiki ICMC - Http://wiki.icmc.usp.br
> Programa, Material de Aulas, Critérios de Avaliação,
> Trabalhos Práticos, Datas das Provas, Notas