Linguagem de Programação C
Aula 2

Professor: Rodrigo Rocha

Tipos, Variáveis e Constantes

- Tipos de Dados
- Variáveis
- Constantes
- Introdução a entrada e saída
  - Printf
  - Scanf
- Ferramentas:
  - Dev-C++
  - Code::Blocks

Tipos de Dados do C

- O C tem 5 tipos básicos:
  char, **int**, **float**, **void, double**.

- Os modificadores de tipo do C são quatro:
  signed, **unsigned**, **long** e **short**.

- Ao **float** não se pode aplicar nenhum e ao **double** pode-se aplicar apenas o **long**.
- Os quatro modificadores podem ser aplicados a inteiros.
- A intenção é que **short** e **long** devam prover tamanhos diferentes de inteiros.
- O modificador **unsigned** serve para especificar variáveis sem sinal.

Tipos de Dados em C

Tipo	Num de bits	Formato para leitura com scanf	Intervalo	
			Inicio	Fim
char	8	%c	-128	127
unsigned char	8	%c	0	255
signed char	8	%c	-128	127
int	16	%i	-32.768	32.767
unsigned int	16	%u	0	65.535
signed int	16	%i	-32.768	32.767
short int	16	%hi	-32.768	32.767
unsigned short int	16	%hu	0	65.535
signed short int	16	%hi	-32.768	32.767
long int	32	%li	-2.147.483.648	2.147.483.647
signed long int	32	%li	-2.147.483.648	2.147.483.647
unsigned long int	32	%lu	0	4.294.967.295
float	32	%f	3,4E-38	3.4E+38
double	64	%lf	1,7E-308	1,7E+308
long double	80	%Lf	3,4E-4932	3,4E+4932

Tipos de Dados em C

- **Caracteres (char):** O C trata os caracteres ('a', 'b', 'x', etc.) como sendo variáveis de um *byte* (8 *bit*s).
- Para indicar um caractere de texto usamos apóstrofes.
- Podemos usar um **char** também para armazenar valores numéricos inteiros.

```
#include <stdio.h>
int main() {
  char ch;
  ch='D';
  printf("%c",ch); /* Imprime D */
  printf("%d",ch); /* Imprime 68 */
  return(0);
```

Tipos de Dados em C

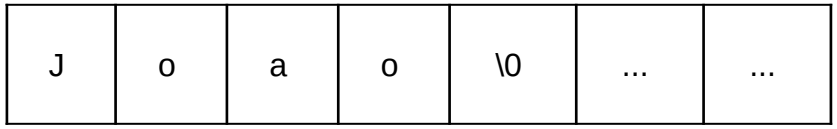
- **Strings :** É um vetor de caracteres terminado com um caractere nulo (código ASCII igual a 0).
- O terminador nulo também pode ser escrito usando a convenção de barra invertida do C como sendo '\0'.
- Para declarar uma string, podemos usar o seguinte formato geral:

  char nome_da_string[tamanho];

- Isto declara um vetor de caracteres (uma string) com número de posições igual a *tamanho*. Note que, como temos que reservar um caractere para ser o terminador nulo, temos que declarar o comprimento da string como sendo, no mínimo, um caractere maior que a maior string que pretendemos armazenar.

Tipos de Dados em C

- Vamos supor que declaremos uma string de 7 posições e coloquemos a palavra João nela. Teremos:



- Para se acessar um determinado caracter de uma string, basta "indexarmos", ou seja, usarmos um índice para acessarmos o caracter desejado dentro da string.
- Suponha uma string chamada *str*. Podemos acessar a segunda letra de *str* da seguinte forma:

```
str[1] = 'a';
```

- Na linguagem C, o índice *começa em zero*.

Variáveis

- A declaração de variáveis em C devem atender as seguintes especificações:
  - O nome deve começar com uma letra ou sublinhado (_) ;
  - Os caracteres subsequentes devem ser letras, números ou sublinhado (_);
  - O nome não pode ser igual a uma palavra reservada;
  - Não pode ser igual ao nome de uma função declarada pelo programador, ou pelas bibliotecas do C.
  - Só pode ter até 32 caracteres.

  - OBS; É bom sempre lembrar que o C é "case sensitive" e portanto deve-se prestar atenção às maiúsculas e minúsculas.

Dicas quanto aos nomes de variáveis...

- É uma prática tradicional do C, usar letras minúsculas para nomes de variáveis e maiúsculas para nomes de constantes. Isto facilita na hora da leitura do código;

- Quando se escreve código usando nomes de variáveis em português, evita-se possíveis conflitos com nomes de rotinas encontrados nas diversas bibliotecas, que são em sua maioria absoluta, palavras em inglês.

Variáveis

- As variáveis no C devem ser declaradas antes de serem usadas. A forma geral da declaração de variáveis é:

  tipo_da_variável lista_de_variáveis;

- As variáveis de uma ***lista de variáveis*** terão todas o mesmo tipo e deverão ser separadas por vírgula.

- Onde declarar:

  Variáveis globais - Fora de todas as funções do programa.
  Variáveis locais - No início de um bloco de código.
  Parâmetros - Na lista de parâmetros de uma função.

Variáveis - Exemplo

```
#include <stdio.h>

int contador;
int func1(int j) {
  /* aqui viria o código da funcao
     ...
  */
}

int main() {
  char condicao;
  int i;
  for (i=0; i<100; i=i+1) { /* Bloco do for */
    float f2;
    /* etc ...
       ...
    */
    func1(i);
  }
  /* etc  ...  */
  return(0);
}
```

Variáveis - Inicialização

- Podemos inicializar variáveis no momento de sua declaração. Para fazer isto podemos usar a forma geral é

  tipo_da_variável nome_da_variável = constante;

- Uma variável ***não*** inicializada, possui um valor ***indefinido*** e que não pode ser utilizado para nada. ***Nunca*** presuma que uma variável declarada vale zero ou qualquer outro valor.

- Exemplos de inicialização são dados abaixo:

```
char ch='D';
int count=0;
float pi=3.141;
```

Constantes

- Constantes são valores que são mantidos fixos pelo compilador.

- São consideradas constantes, por exemplo, os números e caracteres como 45.65 ou 'n', etc...

- Tipos de constantes em C:

Constantes dos tipos básicos
Constantes hexadecimais e octais
Constantes strings
Constantes de barra invertida

Constantes dos tipos básicos

Tipo de Dado	Exemplos de Constantes
Char	'b' '\n' '\0'
Int	2 32000 -130
long int	100000 -467
short int	100 -30
unsigned int	50000 35678
float	0.0 23.7 -12.3e-10
double	12546354334.0 -0.0000034236556

Constantes hexadecimais e octais

- O C permite que se faça isto. As constantes hexadecimais começam com **0x**. As constantes octais começam em **0**.

Constante	Tipo
0xEF	Constante Hexadecimal (8 bits)
0x12A4	Constante Hexadecimal (16 bits)
03212	Constante Octal (12 bits)
034215432	Constante Octal (24 bits)

Constantes strings

- Já mostramos como o C trata strings. Vamos agora alertar para o fato de que uma string **"Joao"** é na realidade uma constante string.

- Isto implica, por exemplo, no fato de que **'t'** é diferente de **"t"**, pois **'t'** é um **char** enquanto que **"t"** é uma constante string com dois **chars** onde o primeiro é **'t'** e o segundo é **'\0'**.

Constantes de barra invertida

- O C utiliza, para nos facilitar a tarefa de programar, vários códigos chamados códigos de barra invertida

Código	Significado
\b	Retrocesso ("back")
\f	Alimentação de formulário ("form feed")
\n	Nova linha ("new line")
\t	Tabulação horizontal ("tab")
\"	Aspas
\'	Apóstrofo
\0	Nulo (0 em decimal)
\\	Barra invertida
\v	Tabulação vertical
\a	Sinal sonoro ("beep")
\N	Constante octal (N é o valor da constante)
\xN	Constante hexadecimal (N é o valor da constante)

Constantes de Pré-Compilador

- Constantes definidas através da diretiva #define são substituídas pelo valor associado em tempo de compilação

Exemplo:

```
#define max 100

int vet[max];

for (int i=0; i<max; i++)
  vet[i]=0;
```

Função: printf

- A função **printf()** tem a seguinte forma geral:

printf (string_de_controle,lista_de_argumentos);

- Teremos, na string de controle, uma descrição de tudo que a função vai colocar na tela. A string de controle mostra não apenas os caracteres que devem ser colocados na tela, mas também quais as variáveis e suas respectivas posições.

Código	Significado
%d	Inteiro
%f	Float
%c	Caractere
%s	String
%%	Coloca na tela um %

Função: printf - Exemplo

- printf("Teste %% %%")
  - "Teste % %"
- printf("%f",40.345)
  - "40.345"
- printf("Um caractere %c e um inteiro %d",'D',120)
  - "Um caractere D e um inteiro 120"
- printf("%s e um exemplo","Este")
  - "Este e um exemplo"
- printf("%s%d%%","Juros de ",10)
  - "Juros de 10%"

Função: scanf

- O formato geral da função **scanf()** é:

  scanf (string-de-controle,lista-de-argumentos);

- Usando a função **scanf()** podemos pedir dados ao usuário.

- Mais uma vez, devemos ficar atentos a fim de colocar o mesmo número de argumentos que o de códigos de controle na string de controle.

Função: scanf - Exemplo

- scanf(“%i“, &i)
  - Lê um inteiro
- scanf("%i%i%i",&i,&j,&k);
  - Lê três inteiros
- scanf(“%c”, &c);
  - Lê um caracter
- scanf("%s”, &s);
  - Lê uma string

EXERCÍCIO

- Elaborar um programa para ler e escrever:
  - Um inteiro
  - Um número real
  - Um caracter
  - Uma string
- Elaborar um programa para ler um caracter e escrever seu valor numérico (ASCII) correspondente.