

---

# Laboratório de Introdução à Ciência da Computação I

## Aula 1 - Estrutura Sequencial

Professores:

Vanderlei Bonato (responsável) - [vbonato@icmc.usp.br](mailto:vbonato@icmc.usp.br)

Luiz Henrique Kiehn (aluno PAE) - [lhkiehn@icmc.usp.br](mailto:lhkiehn@icmc.usp.br)

# Sumário

---

- Estrutura de programas (sequenciais)
- Tipos de dados simples
- Declaração de variáveis
- Entrada/Saída (E/S)
- Operadores e funções pré-definidas
- Exercícios

# A linguagem C

---

- Linguagem de propósito geral conhecida por ser eficiente, econômica e portátil
- Padrão ANSI C
  - ANSI - American National Standards Institute
- Alguns compiladores que suportam ANSI C
  - GCC (GNU Compiler)
  - Microsoft Visual C/C++ Compiler
  - Borland C++ Compiler
  - ...

# Estrutura de programas em C

---

- Um programa em C é uma coleção de diretivas, pragmas, declarações, definições, blocos de comandos/instruções e funções
- Veja as diferenças em:
  - [http://msdn.microsoft.com/en-us/library/aa315887\(VS.60\).aspx](http://msdn.microsoft.com/en-us/library/aa315887(VS.60).aspx)
- Um programa pode ser dividido em um ou mais arquivos fontes
  - É necessário compilar cada arquivo fonte e fazer o “link” dos arquivos objetos resultantes para tornar um programa executável
- Constantes e macros são normalmente organizadas em arquivos separados conhecidos como “header files” ou “include files” que podem ser referenciados a partir de arquivos fontes

# Estrutura de um programa sequencial básico em C

---

```
#include <nome_da_biblioteca>
void main()
{
    instrução 1;
    instrução 2;

    instrução n;
}
```

- As instruções são executadas sequencialmente a partir da função main(), até que uma instrução de desvio ou de retorno seja encontrada

# Declaração de variáveis

---

- O que são variáveis?
  - São referências a áreas de memória do computador que armazenam dados de interesse do programador, dados esses que podem ser alterados a qualquer momento
- A declaração de variáveis é definida pelo programador de acordo com a necessidade para a resolução do problema
- Cada variável é organizada de forma a se garantir a integridade dos dados que contém conforme o tipo definido para ela

# Declaração de variáveis

---

- Variáveis são declaradas após a especificação de seus tipos:
  - type-specifier variable-names
  - Ex:

```
int x;  
int x, y;  
char sexo;  
char nome[40];
```
- A linguagem C possui cinco tipos básicos:
  - int, float, double, void, char

# Nome de variáveis

---

- Podem ter até 32 caracteres
- Devem começar com letra ou sublinhado (\_), sendo que os caracteres subsequentes devem ser letras, números ou sublinhado (\_)
- Não podem coincidir com nomes de palavras reservadas, nem de funções declaradas pelo programador ou em bibliotecas do C
- C é "case sensitive", ou seja, maiúsculas são distintas de minúsculas (Nome  $\neq$  nome  $\neq$  NOME  $\neq$  NoMe)



# A partir dos tipos básicos são definidos outros tipos

---

Tipo	Tamanho (bytes)	Abrangência dos Valores	
char	1	-128	a 127
unsigned char	1	0	a 255
int	2	-32768	a 32767
unsigned int	2	0	a 65535
short int	2	-32768	a 32767
long int	4	-2.147.483.648	a 2.147.483.647
unsigned long int	4	0	a 4.294.967.295
float	4	$\pm 3,4 \cdot 10^{-38}$	a $\pm 3,4 \cdot 10^{38}$
double	8	$\pm 1,7 \cdot 10^{-308}$	a $\pm 1,7 \cdot 10^{308}$
long double	10	$\pm 3,4 \cdot 10^{-4932}$	a $\pm 3,4 \cdot 10^{4932}$

Faixa (*range*) de acordo com o padrão ANSI (considerada mínima)

# O tamanho e faixa de valores podem variar de acordo com o compilador ou processador

---

Tipo	Tamanho (bytes)	Abrangência dos Valores	
char	1	-128	a 127
unsigned char	1	0	a 255
int	4	-2.147.483.648	a 2.147.483.647
unsigned int	4	0	a 4.294.967.295
short int	2	-32768	a 32767
long int	4	-2.147.483.648	a 2.147.483.647
unsigned long int	4	0	a 4.294.967.295
float	4	$\pm 3,4 \cdot 10^{-38}$	a $\pm 3,4 \cdot 10^{38}$
double	8	$\pm 1,7 \cdot 10^{-308}$	a $\pm 1,7 \cdot 10^{308}$
long double	10	$\pm 3,4 \cdot 10^{-4932}$	a $\pm 3,4 \cdot 10^{4932}$

# Comando de atribuição

---

- Para atribuir um conteúdo a uma variável, utiliza-se a seguinte instrução (note o sinal de igualdade):

`<variável> = <conteúdo>`

- O conteúdo atribuído à variável pode ser uma literal (número ou caracter) ou outra variável, e seu tipo deve ser compatível com o tipo da variável

- Algumas variações nesse formato são admitidas  
Por exemplo:

`<variável1> = <variável2> = ... = <conteúdo>;`

# Comando de atribuição

---

- Exemplo de atribuições de valores ou operações a variáveis (sinal de igualdade)

```
x = 4;
```

```
x = x + 2;
```

```
y = 2.5;
```

```
sexo = 'F';
```

- Em C um caracter é representado entre apóstrofos e uma cadeia de caracteres entre aspas
- Para armazenar uma cadeia de caracteres numa variável deve-se utilizar uma função para manipulação dos mesmos, como:  

```
strcpy(nome, "Joao");
```

# Variável global x local

---

- A variável global é visível em todo o programa, enquanto que a local é visível somente dentro da função onde foi declarada

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

```
int x=1;           // x e' uma variavel declarada globalmente
```

```
int main()
```

```
{
```

```
    int y;         // y e' uma variavel declarada localmente
```

```
    y = 2 * x;
```

```
}
```

# Comandos de entrada e saída básicos

---

- Comando de entrada recebe dados digitados pelo usuário e de saída mostra os dados na tela
- Comandos de entrada mais utilizados:

```
/*armazena um ou mais caracteres na variável "nome" */  
gets(nome);
```

```
/*armazena um valor em x) */  
scanf(&x);
```

- Comando de saída

```
/*mostra o número inteiro armazenado na variável "x")  
printf ("%d", x);
```

# Exemplo com o scanf e printf

---

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

void main()
{
    /* Declaracao de variaveis int, float e char */
    int parcela_1, parcela_2, resultado_a;
    char oper;

    /* E/S - solicitacao de dados digitados no teclado */
    printf("Informe as parcelas 1 e 2: ");
    scanf("%d %d", &parcela_1, &parcela_2);

    /* E/S - apresentacao do resultado da soma */
    oper = '+';
    resultado_a = parcela_1 + parcela_2;
    printf("%d %c %d = %d \n", parcela_1, oper, parcela_2, resultado_a);
    system("PAUSE");
}
```

# Principais operadores matemáticos

---

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



# Operadores matemáticos de atribuição

Operador	Exemplo	Comentário
<code>+=</code>	<code>x += y</code>	Equivale a $X = X + Y$ .
<code>-=</code>	<code>x -= y</code>	Equivale a $X = X - Y$ .
<code>*=</code>	<code>x *= y</code>	Equivale a $X = X * Y$ .
<code>/=</code>	<code>x /= y</code>	Equivale a $X = X / Y$ .
<code>%=</code>	<code>x %= y</code>	Equivale a $X = X \% Y$ .
<code>++</code>	<code>x++</code>	Equivale a $X = X + 1$ .
<code>++</code>	<code>y = ++x</code>	Equivale a $X = X + 1$ e depois $Y = X$ .
<code>++</code>	<code>y = x++</code>	Equivale a $Y = X$ e depois $X = X + 1$ .
<code>--</code>	<code>x--</code>	Equivale a $X = X - 1$ .
<code>--</code>	<code>y = --x</code>	Equivale a $X = X - 1$ e depois $Y = X$ .
<code>--</code>	<code>y = x--</code>	Equivale a $Y = X$ e depois $X = X - 1$ .

# Exemplos de expressões válidas

---

int x, y, z;

x=y=z=15; /\* ⇔ x=15; y=15; z=15; \*/

x=x+2; /\* ⇔ x passa a valer seu valor anterior + 2 \*/

x++; /\* ⇔ x=x+1; \*/

x=++y; /\* ⇔ y=y+1; x=y; \*/

x=y++; /\* ⇔ x=y; y=y+1; \*/

x=-x; /\* ⇔ x = (-1).x \*/

x=x+y-(z--); /\* ⇔ x=x+y-z; z=z-1; \*/

x+=y; /\* ⇔ x=x+y; \*/

x=(++y,y+2); /\* ⇔ y=y+1; x=y+2);

x=(y++,y+2); /\* ⇔ y=y+1; x=y+2);

# Exercício I

---

Considere as seguintes variáveis:

- `int i, j, k;`
- `float x, y, z;`

Responda:

- Se `x=12.` e `y=15.`, quanto vale `z = y / x`?
- Se `i=12` e `y=15.`, quanto vale `z = y / i`?
- Se `i=12` e `j=15`, quanto vale `z = j / i`?
- Se `i=12` e `j=15`, quanto vale `z = (float) j / i`?
- Se `i=10` e `j=2`, quanto valem `i, j` e `k` após calculada a expressão..... `k = i++ - --j`?
- Se `i=10` e `j=2`, quanto valem `i` e `j` após calculada a expressão..... `i /= j`?
- Se `x=2.71` e `y=3.2`, quanto valem `x` e `y` após calculada a expressão.... `x += (y++, y / 2)`?

# Referências

---

Ascencio AFG, Campos EAV. Fundamentos de programação de computadores. São Paulo : Pearson Prentice Hall, 2006. 385 p.

Material do Prof. Dr. Fernando Santos Osório do ICMC/USP com teoria e lista de exercícios:

<http://coteia.icmc.usp.br/mostra.php?ident=624>

Outras fontes interessantes

<http://mtm.ufsc.br/~azeredo/cursoC/c.html>

<http://www.cs.cf.ac.uk/Dave/C/>

[http://msdn.microsoft.com/en-us/library/aa315845\(VS.60\).aspx](http://msdn.microsoft.com/en-us/library/aa315845(VS.60).aspx)

[http://www.acm.uiuc.edu/webmonkeys/book/c\\_guide/](http://www.acm.uiuc.edu/webmonkeys/book/c_guide/)

<http://techpubs.sgi.com/library/manuals/0000/007-0701-150/pdf/007-0701-150.pdf>