Aula 6 – Tutorial Rápido de C

1. Introdução

Nesta aula será apresentada a linguagem C, relacionando-a com o que já aprendemos de pseudocódigo. É preciso, no entanto, avisar:

- Para explicar porque algumas coisas são do jeito que são em C é preciso entender conceitos mais avançados, como *strings*, ponteiros, etc. Portanto, alguns conceitos podem parecer desnecessariamente complicados. Porém devemos usar a linguagem do jeito que ela foi definida.

2. Criação de Programas

```
Algoritmo MeuPrograma
VAR
     ...
Início
     ...
Fim.
```

```
main() {
     ...
}
```

- Comentários podem ser escritos entre `/*` e `*/` ou utilizando `//`.
- C é sensível ao caso (*case sensitive*);
- Todo comando termina com "`;`" em C.

3. Tipos de Dados e Declaração de Variáveis

Pseudocódigo	**C**
Inteiro	`int`
Real	`float`
Literal	`char`
Lógico	`int`

```
VAR
a, b, c: inteiro
x, y: real
nome: literal[50]
pagou: lógico
```

```
int a, b, c;
float x, y;
char nome[50];
int pagou;
```

Note que:

- É preciso especificar o número de caracteres em uma variável do tipo `char`;
- Não há tipo lógico (booleano) em C. Usa-se um inteiro, onde `0` é `.F.` e qualquer outro valor é `.V.`;
- Nomes de variáveis seguem a mesma regra do pseudocódigo: devem começar com uma letra ou sublinhado ("_"), seguido de letras, sublinhados e/ou números;
- Existem outros tipos (`short`, `long`, `double`, `long double`, o modificador `signed`/`unsigned`, etc.). Porém aqui vamos simplificar.

4. Expressões

Pseudo	C		Pseudo	C
+	+		.V.	1
-	-		.F.	0
*	*		.OU.	||
/	/		.E.	&&
**	pow()		.NÃO.	!
=	==		(	(
<>	!=		)	)
<	<			
<=	<=		**Literal**	
>	>		Ver seção 5.	
>=	>=			

```
(a + (b - c) * 4) / pow(2, 10)

(a == b) || (! (b == c))

pagou && maiorDeIdade

(x > y) && (z < w)

(resposta != 10)

// Equivalente a c = c + 1;
c++;

// Equivalente a d = d - 1;
d--;
```

Note que:

- `pow()` requer #include `<math.h>`;
- A divisão entre inteiros retorna um resultado inteiro (descarta o resto);
- Existe um operador também para o resto da divisão inteira: `%`;
- Existem operadores de incremento (++) e decremento (--).

5. Strings

- *Strings* são vetores de caracteres (veremos mais sobre vetores em uma aula futura);
- O tipo `char`, sem especificar tamanho, determina um caractere único;
  - Note a diferença entre `"a"` e `'a'`.
- Funções:
  - Para ler uma *string*, usa-se `gets(str);`
  - Para atribuir uma *string* a outra, usa-se `strcpy(strDestino, strOrigem);`
  - Para concatenar uma *string* a outra, usa-se `strcat(strDestino, strOrigem);`
  - Para saber o tamanho de uma *string*, usa-se `strlen(str);`
  - Para comparar uma *string* com outra, usa-se `strcmp(str1, str2)`: retorna 0 se iguais, < 0 se `str1 < str2`, > 0 se `str1 > str2`;
- Todas as funções acima requerem `#include <string.h>`.

6. Instruções Primitivas

Pseudocódigo	C
<-	=
Leia	gets(), scanf()
Escreva	printf()

```
a = a + 10;

/* Operadores podem ser combinados
   com a atribuição. */
a += 10;

gets(nome); // Só para strings!
printf("Olá, mundo!");
```

Note que:

- As funções acima requerem `#include <stdio.h>`;
- Atribuição pode ser encadeada: `x = y = z = 2.5`.

6.1. Concatenando dados para impressão com `printf`:

- O `printf()` tem um argumento obrigatório: uma *string*;
- Após esta *string*, você pode incluir outros dados para impressão, de diversos tipos diferentes;
- Tais dados são mesclados na primeira *string*, obedecendo códigos de impressão (estes são os mais básicos, para simplificar):
  - `%d` é substituído por um inteiro (`int`);
  - `%f` é substituído por um real (`float`);
  - `%c` é substituído por um caractere (único);
  - `%s` é substituído por um literal (vários caracteres, *string*);
  - `%%` é substituído pelo caractere `%`.
- O `printf()` não quebra linha automaticamente. É preciso adicionar os caracteres `\n` ao final do primeiro argumento da função.

6.2. Lendo dados de tipos diversos com `scanf`:

- Funciona como um `printf()` ao contrário, utilizando os mesmos códigos acima para determinar o tipo de dado que está sendo lido;
- Portanto, o primeiro parâmetro é uma *string* especificando um dos códigos, o segundo parâmetro é a variável onde será armazenado o dado lido;
- Ao contrário de `printf()`, porém, ao especificar as variáveis nas quais o dado será armazenado, é preciso utilizar o caractere `&` antes do nome das variáveis;
- Para *strings*, no entanto, use `gets()`, pois é mais simples. Pode-se usar o `scanf()` com `%s`, porém para strings não se usa o caractere `&`!

```
#include <stdio.h>
main() {
    char nome[50];
    int idade;
    float valor;
    gets(nome);           // Ou scanf("%s", nome); <- sem o &.
    scanf("%d", &idade);
    scanf("%f", &valor);
    printf("%s, %d, %f\n", nome, idade, valor);
}
```

7. Controle de Fluxo

7.1. Blocos

- Onde encontrarmos <Comando ou bloco>, podemos substituir por um único comando ou um bloco de comandos. Um bloco começa com `{` e termina com `}`.

7.2. Se

```
Se <Condição> Então
     <Conjunto de Comandos 1>
Senão
     <Conjunto de Comandos 2>
Fim_se
```

```
Se <Condição> Então
     <Conjunto de Comandos>
Fim_se
```

```
if (<Condição>)
     <Comando ou bloco 1>
else
     <Comando ou bloco 2>;
```

```
if (<Condição>)
     <Comando ou bloco>;
```

```
#include <stdio.h>
main() {
     int idade;
     scanf("%d", &idade);
     if (idade < 18) printf("Menor de idade.\n");
     else printf("Maior de idade.\n");
}
```

7.3. Escolha

```
Escolha
Caso <Condição 1>
     <Conjunto de Comandos 1>
Caso <Condição 2>
     <Conjunto de Comandos 2>

...

Caso <Condição n>
     <Conjunto de Comandos n>
Senão
     <Conjunto de Comandos Senão>
Fim_escolha
```

```
switch (<variável>) {
case <valor 1>:
     <Comando ou bloco 1>
     break;

case <valor 2>:
     <Comando ou bloco 2>
     break;
...

default:
     <Comando ou bloco senão>
}
```

```
#include <stdio.h>
main() {
     char resposta;
     printf("Digite S para sim e N para não: ");
     scanf("%c", &resposta);

     switch (resposta) {
     case 'S':
     case 's':
          printf("Você respondeu sim.\n");
          break;

     case 'N':
     case 'n':
          printf("Você respondeu não.\n");
          break;

     default:
          printf("Você respondeu algo diferente.\n");
     }
}
```

Note que:

- O `switch` só permite comparar valor de variáveis discretas. Para condições diferentes, usar `if-else-if` encadeados.

7.4. Para

```
Para <var> de <início> até <final> incr de <inc> faça
     <Conjunto de Comandos>
Fim_para


// Para decremento, usar <var>-- e a comparar >= <final>.
// Para incrementos diferentes, usar <var> += <incr>
for (<var> = <início>; <var> <= <final>; <var>++)
     <Conjunto de Comandos>


#include <stdio.h>
main() {
     int i;
     for (i = 0; i <= 10; i++)
          printf("%d\n", i);
}
```

7.5. Enquanto

```
Enquanto <Condição> faça
     <Conjunto de Comandos>
Fim_enquanto
```

```
while (<Condição>)
     <Conjunto de Comandos>
```

```
#include <stdio.h>
main() {
     int i = 0;
     while (i <= 10) {
          printf("%d\n", i);
          i++;
     }
}
```

7.6. Repita

```
Repita
     <Conjunto de Comandos>
Até que <Condição>
```

```
do
     <Conjunto de Comandos>
while (<Condição>);
```

```
#include <stdio.h>
main() {
     int i = 0;
     do {
          printf("%d\n", i);
          i++;
     } while (i <= 10);
}
```

8. Informações mais aprofundadas sobre a linguagem C

8.1. Tabela de tipos de dados completa

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

8.2. Tabela de precedência

Maior precedência

```
() [] ->
! ~ ++ -- . -(unário)
(cast) *(unário)
&(unário) sizeof
* / %
+ -
<< >>
<<= >>=
== !=
&
^
|
&&
||
?
= += -= *= /=
```

Menor precedência

```
,
```

8.3. Coerção entre tipos

- Sintaxe: (<tipo>) <valor ou variável>
- Exemplos:
  - `int i =(int)2.5;`
  - `float f = (float)variavelInteira;`

8.4. O operador ternário

- Sintaxe: `<variável> = (<condição>) ? <valor 1> : <valor 2>;`
- Semântica: se `<condição>` for verdadeira, atribua `<valor 1>` à `<variável>`, do contrário atribua `<valor 2>`.

8.5. Outros comandos

- O comando `break` pode sair de um laço (além do `switch`, que já vimos);
- O comando `continue` pode continuar um laço;
- Um comando `goto` pode ir para uma linha específica do programa (identificada por um rótulo).

8.6. Funções úteis já definidas

- `#include <stdlib.h>`:
  - `abs(valorInteiro)`: retorna o valor absoluto (ignora sinal);
  - `rand()`: retorna um número pseudoaleatório;
  - `srand(valorInteiroPositivo)`: define a semente para a geração de números pseudoaletórios;
- `#include <math.h>`:
  - `sin(valorReal)`, `cos(valorReal)`, `tan(valorReal)`: funções trigonométricas;
  - `sqrt(valorReal)`: raiz quadrada;
  - `exp(valorReal)`: exponencial (e^x);
  - `log(valorReal)`: logaritmo com base natural ($\log_e$);
  - `log10(valorReal)`: logaritmo com base 10 ($\log_{10}$);
  - `pow(valorReal1, valorReal2)`: exponenciação;
  - `ceil(valorReal)`: teto (arredondar para cima);
  - `floor(valorReal)`: piso (arredondar para baixo);
  - `fabs(valorReal)`: valor absoluto (igual a abs(), só que para valores reais).
- Dentre muitas outras bibliotecas...

Exercícios – Tutorial Rápido de C

(Os mesmos exercícios da Aula 5)

1) Escreva um algoritmo para determinar o maior entre dois números reais dados [Saliba, 1992, p. 72].

2) Uma empresa decide dar um reajuste a seus funcionários de acordo com o seguinte critério: 50% de aumento para os que ganham menos de R$ 3.000,00, 20% para os que ganham entre R$ 3.000,00 e R$ 10.000,00 (inclusive) e 15% para os demais. Escreva um algoritmo para calcular este reajuste [Saliba, 1992, p. 74].

3) Escreva um algoritmo para calcular a soma de 10 números quaisquer fornecidos pelo usuário [Saliba, 1992, p. 77].

4) Escreva um algoritmo para calcular os n primeiros termos de uma progressão aritmética, dados o elemento inicial a0 e a razão r.

5) Faça um algoritmo para imprimir os n primeiros termos da sequência de Fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, ...

6) Uma empresa decide presentear seus funcionários com um bônus de Natal, cujo valor é definido pelos critérios a seguir. Elabore um algoritmo para calcular o valor do bônus concedido a cada um dos 50 funcionários e o impacto de tal atitude no orçamento da empresa.

- Os funcionários do sexo masculino com tempo de casa superior a 15 anos terão direito a um bônus de 20% do seu salário;
- As funcionárias com tempo de casa superior a 10 anos terão direito a um bônus de 25% do seu salário;
- Os demais funcionários terão direito a um bônus de R$ 5.000,00.

7) Escreva um algoritmo para calcular o fatorial de um número inteiro.

Resolução dos Exercícios – Controle do Fluxo de Execução

1)
```
#include <stdio.h>
main() {
     float a, b;
     printf("Digite dois números reais:\n");
     scanf("%f", &a);
     scanf("%f", &b);
     if (a > b)
          printf("O maior é: %f\n", a);
     else
          printf("O maior é: %f\n", b);
}
```

2)
```
#include <stdio.h>
main() {
     float salario;
     printf("Salário = ");
     scanf("%f", &salario);

     if (salario < 3000)
          salario = salario * 1.5;
     else if (salario < 10000)
          salario = 1.2 * salario;
     else
          salario *= 1.15;

     printf("Salário reajustado = %f\n", salario);
}
```

3)
```
#include <stdio.h>
main() {
     float numero, soma = 0.0;
     int i;

     for (i = 1; i <= 10; i++) {
          printf("Escreva o %dº número: ", i);
          scanf("%f", &numero);
          soma += numero;
     }
     printf("Soma = %f\n", soma);
}
```

4)

```
#include <stdio.h>
main() {
     float a0, r;
     int i, n;

     printf("Informe a0, r e n.\n");
     scanf("%f", &a0);
     scanf("%f", &r);
     scanf("%d", &n);

     for (i = 0; i < n; i++) {
          printf("%f\n", a0);
          a0 += r;
     }
}
```

5)

```
#include <stdio.h>
main() {
     int ant1, ant2, i, n, x;

     printf("Quantos números de Fibonacci você quer? ");
     scanf("%d", &n);
     ant1 = 1;
     ant2 = 0;

     if (n > 0) printf("0, ");
     if (n > 1) printf("1, ");

     for (i = 3; i <= n; i++) {
          x = ant1 + ant2;
          printf("%d, ", x);
          ant2 = ant1;
          ant1 = x;
     }
     printf("\n");
}
```

6)

```
#include <stdio.h>
main() {
     char sexo, nome[15];
     int i, tempoCasa;
     float salario, bonus, impacto = 0;

     for (i = 0; i < 50; i++) {
          printf("Nome? ");
          gets(nome);
          printf("Sexo (M/F)? ");
          scanf("%c", &sexo);
          printf("Tempo de casa (em anos)? ");
          scanf("%d", &tempoCasa);
          printf("Salário (em R$)? ");
          scanf("%f", &salario);

          if ((sexo == 'M') && (tempoCasa >= 15))
               bonus = salario * 0.2;
          else if ((sexo == 'F') && (tempoCasa >= 10))
               bonus = salario * 0.25;
          else
               bonus = 5000;

          impacto += bonus;
          printf("%s terá bônus de R$ %f\n", nome, bonus);
     }

     printf("O impacto no orçamento é de R$ %f\n", impacto);
}
```

7)

```
#include <stdio.h>
main() {
     int n, original, fat = 1;
     printf("Digite n: ");
     scanf("%d", &n);
     original = n;

     while (n > 0) {
          fat = fat * n;
          n--;
     }
     printf("%d! = %d\n", original, fat);
}
```