

Algoritmos e Estruturas de Dados I

Aula 3 -Estrutura Sequencial, variáveis, entrada e saída

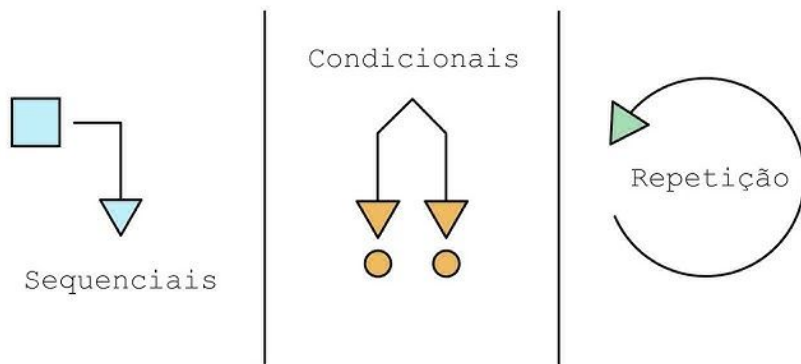
Prof. Felipe Lara



Estrutura Sequencial

Existem três estruturas básicas em programação:

- **Estrutura Sequencial**
- Estrutura Condicional
- Estrutura Repetitiva



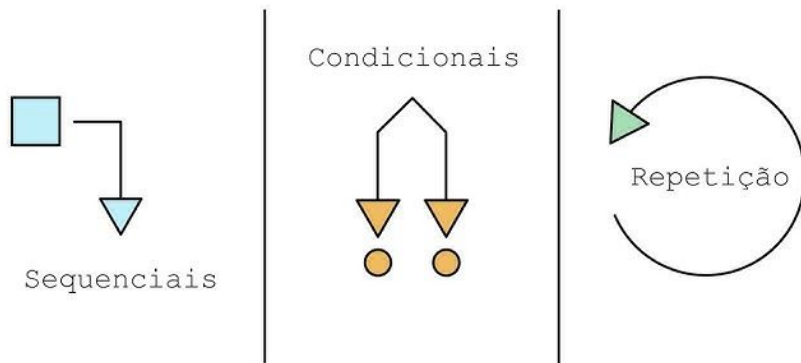
Um conjunto de instruções em que cada instrução é executada em ordem, obedecendo a uma lógica de programação.

Exemplo de problema: calcule a média de 3 notas

Estrutura Sequencial

Existem três estruturas básicas em programação:

- Estrutura Sequencial
- **Estrutura Condicional**
- Estrutura Repetitiva



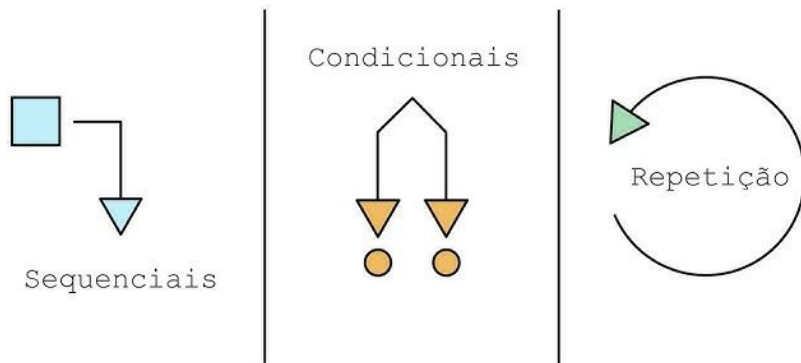
Permite a tomada de decisões com base em condições específicas, possibilitando a execução de diferentes blocos de código.

Exemplo de problema: calcule a média de 3 notas **e verifique se a média é maior que 60**

Estrutura Sequencial

Existem três estruturas básicas em programação:

- Estrutura Sequencial
- Estrutura Condicional
- **Estrutura Repetitiva**



Permite a execução de um bloco de código múltiplas vezes, de acordo com uma condição ou um contador.

Exemplo de problema: calcule a média da nota de uma turma de **100 alunos**

Estrutura Sequencial



Estrutura Sequencial em C

```
#include <nome da biblioteca>
```

```
int main()
```

```
{
```

```
    bloco de comandos;
```

```
}
```

Estrutura Sequencial

- Declaração de Variáveis: as variáveis são declaradas no início do programa, permitindo o armazenamento e manipulação de dados durante a execução
- Recebimento de Entrada: os dados são recebidos do usuário por meio de comandos de entrada para serem utilizados nos cálculos e operações.
- Processamento e Saída: as operações são realizadas de forma sequencial, e os resultados são exibidos para o usuário por meio de comandos de saída

Estrutura Sequencial

- **Declaração de Variáveis:** as variáveis são declaradas no início do programa, permitindo o armazenamento e manipulação de dados durante a execução
- Recebimento de Entrada: os dados são recebidos do usuário por meio de comandos de entrada para serem utilizados nos cálculos e operações.
- Processamento e Saída: as operações são realizadas de forma sequencial, e os resultados são exibidos para o usuário por meio de comandos de saída

Variável



Dados

- **Dado** - toda e qualquer tipo de informação manipulada por um programa
- **Tipo de Dado** - representação computacional da informação. Define como armazená-la e como interpretá-la.
 - Em computação, uma estrutura de dados é uma forma de armazenar e organizar os dados de modo que eles possam ser usados de forma eficiente.
 - **Tipos Básicos** - mais utilizados. Podem ser numéricos, lógicos, literais ou caracteres.

Tipos Básicos de Dados

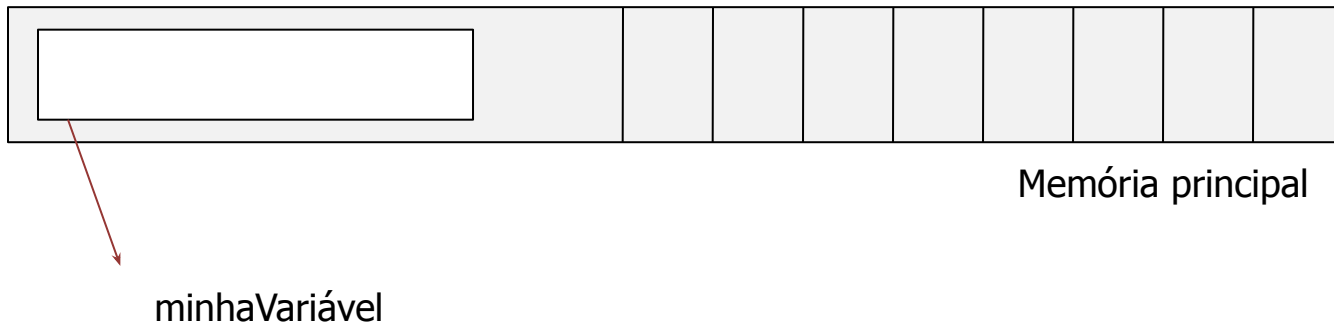
- **Numérico:** 2, 10, 381, 584, -15, -78 (sem vírgula)
- **Numérico:** 3.141592653, 2.71828 (com vírgula)
- **Lógico:** True, False
- **Caracteres:** 'z', 'f', '1'
- **Texto:** "Exemplo", "Eu amo AEDs I"

Variável

- Um programa de computador recebe dados que necessitam ser armazenados no computador para serem utilizados no processamento

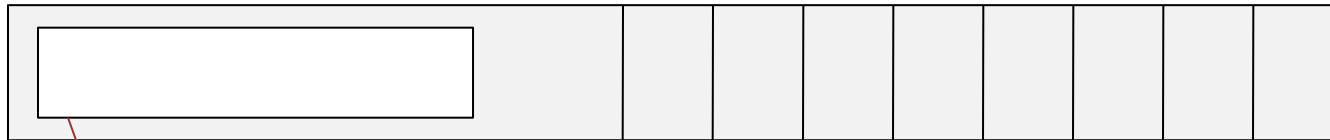
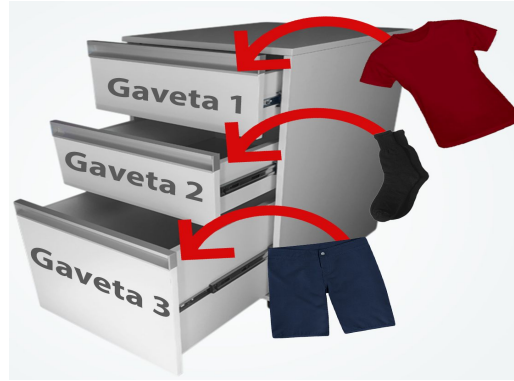
Variável

- Um programa de computador recebe dados que necessitam ser armazenados no computador para serem utilizados no processamento
- Variável: representa um espaço na memória
 - Nome e Tipo
 - Conteúdo pode variar durante a execução
 - Deve ser declaradas no início do algoritmo



Variável

- Analogia com gavetas



Memória principal

minhaVariável

Variável

Variável - local de armazenamento reservado na memória RAM (volátil) de um computador em tempo de execução, associado a um tipo e a um nome.

- Com o término do programa, a variável é desalocada da memória.
- Toda variável tem um nome, um tipo e um valor.
- O nome será escolhido pelo desenvolvedor/programador e deve atender a algumas regras. Sugere-se que comece com minúscula
 - Ex: idade, nome, idadeAluno

Tipos de Dados:

Tipo Primitivo	Descrição
Inteiro	Representa o conjunto de números inteiros
Real	Representa o conjunto de números reais
Caractere	Representa um ou mais caracteres do teclado
Lógico	Representa um valor lógico (V ou F)

Declaração de Variável

Declaração de Variável – o comando que efetivamente “cria” a variável, reservando-a em memória e associando nome e tipo a ela.

O local onde a variável foi armazenada é identificado internamente no computador pelo endereço de memória.

Inicialmente, não vamos precisar nos preocupar com a alocação de memória; ela será feita de maneira “automática”.

Atribuição – armazena dado na variável, substituindo conteúdo anterior pelo indicado.

Tipos Básicos de Dados

- Numérico: 2, 10, 381, 584, -15, -78 (sem vírgula) -> **int**
- Numérico: 3.141592653, 2.71828 (com vírgula) -> **float**
- Lógico: True, False -> **bool**
- Caracteres: 'z', 'f', '1' -> **char**
- Texto: "Exemplo", "Eu amo AEDs I" -> **string**

Declaração de Variáveis em C

Tipos básicos:

Tipo	Descrição
int	Utilizado para definir uma variável inteira que comporta valores pertencentes ao conjunto dos números inteiros
float	Utilizado para definir uma variável real que comporta valores pertencentes ao conjunto dos números reais
double	O tipo double é similar ao float , a diferença é que o este tipo comporta valores reais com um número de dígitos maior, assim, a precisão nos cálculos com casas decimais aumenta
char	Utilizado para armazenar um caractere. Comporta apenas um caractere

Declaração de Variáveis em C

As variáveis são declaradas após a especificação de tipo das mesmas

Exemplo:

```
int main()
{
    int Y;
    float X;
    bool W;
    char sexo, nome[40];
}
```

Estrutura Sequencial em C

- C é case sensitive: considera que letras maiúsculas são diferentes de minúsculas (por exemplo, *a* é diferente de *A*) .
 - *Ex.:*
 - *NOTA != nota != Nota....*
- **Comandos** em C são, na maioria das vezes, escritos com letras minúsculas.

Declaração de Variáveis em C

Tipo	Memória (bytes)	Valor Mínimo	Valor Máximo	Formato específico
short int	2	-32.768	32.767	%hd
unsigned short int	2	0	65.535	%hu
unsigned int	4	0	4.294.967.295	%u
int	4	-2.147.483.648	2.147.483.647	%d
long int	4	-2.147.483.648	2.147.483.647	%ld
unsigned long int	4	0	4.294.967.295	%lu
long long int	8	$-(2^{63})$	$(2^{63})-1$	%lld
unsigned long long int	8	0	18.446.744.073.709.551.615	%llu
signed char	1	-128	127	%c
unsigned char	1	0	255	%c
float	4	1,2E-38	3,4E+38	%f
double	8	1,7E-308	1,7E+308	%lf
long double	16	3,4E-4932	1,1E+4932	%Lf

Declaração de Variáveis em C

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Comando de Atribuição em C

Utilizado para atribuir valores ou operação à variáveis.

- Representado por = (sinal de igualdade)

Exemplos:

```
X = 4;
```

```
X = X + 2;
```

```
Sexo = 'F';
```


Estrutura Sequencial

- Declaração de Variáveis: as variáveis são declaradas no início do programa, permitindo o armazenamento e manipulação de dados durante a execução
- **Recebimento de Entrada:** os dados são recebidos do usuário por meio de comandos de entrada para serem utilizados nos cálculos e operações.
- Processamento e Saída: as operações são realizadas de forma sequencial, e os resultados são exibidos para o usuário por meio de comandos de saída

Comando de Entrada



Comando de Entrada em C

Comando de entrada em C

- Comandos mais utilizados: `scanf` e `fgets`

Comando de Entrada em C

- **Função scanf()**

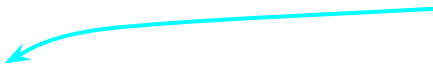
- Permite que os dados digitados pelo usuário sejam armazenados nas variáveis do programa

- **Sintaxe:**

```
scanf("formato", enderecosArgumentos);
```

- **Exemplos:**

```
int mat;  
scanf("%d", &mat);  
float nota1, nota2;  
scanf("%f %f", &nota1, &nota2);
```



O caractere **&** é de uso **obrigatório** e responsável por indicar que o retorno será o **endereço de memória** de uma determinada variável

Comando de Entrada em C

- **Função scanf()**

- Permite que os dados digitados pelo usuário sejam armazenados nas variáveis do programa

- **Sintaxe:**

```
scanf("formato", enderecosArgumentos);
```

- **Exemplos:**

```
int mat;  
scanf("%d", &mat);  
float nota1, nota2;  
scanf("%f %f", &nota1, &nota2);
```

O "%d" indica qual o tipo da variável que está sendo lida.

"%d" para int
"%f" para float

Comando de Entrada em C

- **Função scanf()**

- Permite que os dados digitados pelo usuário sejam armazenados nas variáveis do programa

- **Sintaxe:**

```
scanf("formato", enderecosArgumentos);
```

- **Exemplos:**

```
int mat;  
scanf("%d", &mat);  
float nota1, nota2;  
scanf("%f %f", &nota1, &nota2);
```

Posso ler mais de uma variável ao mesmo tempo. Isso seria igual a:

```
scanf("%f", &nota1);  
scanf("%f", &nota2);
```

Comando de Entrada em C

- **Função scanf()**

- Permite que os dados digitados pelo usuário sejam armazenados nas variáveis do programa

- **Sintaxe:**

```
scanf("formato", enderecosArgumentos);
```

- **Exemplos:**

```
int mat;  
scanf("%d", &mat);  
float nota1, nota2;  
scanf("%f %f", &nota1, &nota2);
```

- **Observação:** a função `scanf()` não é a mais adequada para leitura de textos. Neste caso deve ser substituída pela função `fgets()`

Estrutura Sequencial

- Declaração de Variáveis: as variáveis são declaradas no início do programa, permitindo o armazenamento e manipulação de dados durante a execução
- Recebimento de Entrada: os dados são recebidos do usuário por meio de comandos de entrada para serem utilizados nos cálculos e operações.
- **Processamento e Saída:** as operações são realizadas de forma sequencial, e os resultados são exibidos para o usuário por meio de comandos de saída

Comando de Saída



Comando de Saída em C

Comando de saída em C

- Comandos mais utilizados: `printf` e `puts`

Comando de Saída em C

- **Função printf()**

- Permite realizar a impressão de dados formatados na saída padrão (monitor), ou seja, é responsável pela saída de informações
- Possui um número variado de parâmetros, tantos quantos forem necessários

- **Sintaxe:**

```
printf("formato", argumentos);
```

- **Exemplos:**

```
int mat = 335642;
```

```
float medF = 7;
```

```
printf("Matricula: %d, Med Final: %f ", mat, medF);
```

O valor da variável mat será substituído no texto, no local onde está o %d

O valor da variável medF será substituído no texto, no local onde está o %f

Comando de Saída em C

- **Função printf()**

- Permite realizar a impressão de dados formatados na saída padrão (monitor), ou seja, é responsável pela saída de informações
- Possui um número variado de parâmetros, tantos quantos forem necessários

- **Sintaxe:**

```
printf("formato", argumentos);
```

- **Exemplos:**

```
int mat = 335642;  
float medF = 7;  
printf("Matricula: %d, Med Final: %f ", mat, medF);
```

Resultado:

Matricula: 335642, Med Final: 7

Comando de Saída em C (usando \n)

- **Função printf()**

- Permite realizar a impressão de dados formatados na saída padrão (monitor), ou seja, é responsável pela saída de informações
- Possui um número variado de parâmetros, tantos quantos forem necessários

- **Sintaxe:**

```
printf("formato", argumentos);
```

- **Exemplos:**

```
int mat = 335642;  
float medF = 7;  
printf("Matricula: %d \nMed Final: %f ", mat, medF);
```



O \n usado dentro do printf é responsável por quebrar linha. Resultado:

Matricula: 335642

Med Final: 7

Comando de Saída em C

- **Função puts()**

- Permite realizar a impressão de textos na saída padrão (monitor), ou seja, é responsável pela saída de informações

- **Sintaxe:**

```
puts ("texto");
```

- **Exemplos:**

```
puts ("Digite um numero");
```

Comando de Saída em C

Formato	Tipo da Variável	Conversão Realizada
%c	Caracteres	char, short int, int, long int
%d	Inteiros	int, short int, long int
%e	Ponto flutuante, notação científica	float, double
%f	Ponto flutuante, notação decimal	float, double
%o	Saída em octal	int, short int, long int, char
%s	String	char *, char[]
%u	Inteiro sem sinal	unsigned int, unsigned short int, unsigned long int
%x	Saída em hexadecimal (0 a f)	int, short int, long int, char
%X	Saída em hexadecimal (0 a F)	int, short int, long int, char

Comentários

- Comentários são textos que podem ser inseridos em um programa com o objetivo de documentá-lo e para facilitar o entendimento do mesmo.
 - Lembre-se: o código não é feito somente para você! Futuramente outros poderão necessitar do seu código.
- Os comentários não são analisados pelo compilador.

Comentários

- Comentários são textos que podem ser inseridos em um programa com o objetivo de documentá-lo e para facilitar o entendimento do mesmo.
 - Lembre-se: o código não é feito somente para você! Futuramente outros poderão necessitar do seu código.
- Os comentários não são analisados pelo compilador.
- Os comentários podem ocupar uma ou várias linhas, devendo ser inseridos nos programas utilizando:
 - **`/* */` : comentário de várias linhas**
 - A região de comentários é aberta com os símbolos **`/*`** e é encerrada com os símbolos **`*/`**
 - **`//` : comentário de uma linha**
 - A região de comentários é aberta com os símbolos **`//`** e é encerrada automaticamente ao final da linha

Comentários

Exemplo:

```
/*
```

```
Linhas de comentários...
```

```
Linhas de comentários...
```

```
*/
```

Ou

```
// Linha de comentário
```

Exemplo de código em C

Pseudocódigo

ALGORITMO

DECLARE x, y NUMÉRICO

ESCREVA "Digite um numero."

LEIA x

$y \leftarrow 2 * x$

ESCREVA "O dobro é = ", y

FIM_ALGORITMO

Exemplo de código em C

Pseudocódigo

ALGORITMO

DECLARE x, y NUMÉRICO

ESCREVA "Digite um numero."

LEIA x

$y \leftarrow 2 * x$

ESCREVA "O dobro é = ", y

FIM_ALGORITMO

C

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

```
int main() {
```

```
    int x, y;
```

```
    printf("Digite um numero.\n");
```

```
    scanf("%d", &x);
```

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

```
    printf("O dobro e = %d", y);
```

```
}
```

Exemplo de código em C

Pseudocódigo

ALGORITMO

DECLARE x, y NUMÉRICO

ESCREVA "Digite um numero."

LEIA x

y \square 2*x

ESCREVA "O dobro é = ", y

FIM_ALGORITMO

C

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

```
int main() {
```

```
    int x;
```

```
    printf("Digite um numero.\n");
```

```
    scanf("%d", &x);
```

```
    printf("O dobro e = %d", 2*x);
```

```
}
```

Operadores e Funções Predefinidas

A linguagem C possui operadores e funções predefinidas destinadas a cálculos matemáticos e à manipulação de caracteres.

Operador de Atribuição

Operador	Exemplo	Comentário
=	X = Y	O conteúdo da variável Y é atribuído a variável X.

Operadores Matemáticos

Operador	Exemplo	Comentário
+	$X + Y$	Soma o conteúdo de X e de Y.
-	$X - Y$	Subtrai o conteúdo de Y do conteúdo de X
*	$X * Y$	Multiplica o conteúdo de X pelo conteúdo de Y
/	X / Y	Obtém o quociente da divisão de X por Y
%	$X \% Y$	Obtém o resto da divisão de X por Y
++	$X ++$	Aumenta o conteúdo de X em uma unidade
--	$X --$	Diminui o conteúdo de X em uma unidade

*O operador % só pode ser utilizado com operandos do tipo **inteiro***

Operadores de Atribuição (Matemáticos)

Operador	Exemplo	Comentário
+=	$X \text{ += } Y$	Equivale a $X = X + Y$.
-=	$X \text{ -= } Y$	Equivale a $X = X - Y$.
*=	$X \text{ *= } Y$	Equivale a $X = X * Y$.
/=	$X \text{ /= } Y$	Equivale a $X = X / Y$.
%=	$X \text{ %= } Y$	Equivale a $X = X \% Y$.

Conversão de Tipos

- Conversão de tipos – é a mudança da forma de representação de um tipo para outro
- Conversão implícita – ocorre automaticamente para alguns tipos. Normalmente, refere-se a um subconjunto atribuído a uma variável com tipo que o engloba
- Conversão explícita – necessária quando existe o risco de perder informação (precisão) na conversão.

- Exemplos em C:

```
float aceleracao = 2;  
double velocidade = 5.0;  
int quantidade = (int) velocidade;  
int total = 7.2;
```

Conversão explícita

Converte o valor da variável velocidade para um valor inteiro, ficando igual a 5

Conversão de Tipos

- Conversão de tipos – é a mudança da forma de representação de um tipo para outro
- Conversão implícita – ocorre automaticamente para alguns tipos. Normalmente, refere-se a um subconjunto atribuído a uma variável com tipo que o engloba
- Conversão explícita – necessária quando existe o risco de perder informação (precisão) na conversão.

- Exemplos em C:

```
float aceleracao = 2;  
double velocidade = 5.0;  
int quantidade = (int) velocidade;  
int total = 7.2;
```

Conversão implícita

Converte o valor da variável total para um valor inteiro, ficando igual a 7

Variáveis Constantes

- Constante – informação armazenada em um primeiro momento não pode ser alterada posteriormente
- O nome segue as regras dos identificadores das variáveis. Sugere-se o uso de maiúsculas apenas.
- Exemplos:

```
#define VELOCIDADE 5.0 //PRE-PROCESSADOR, NÃO EXISTE A VARIÁVEL
```

```
const double VELOCIDADE = 5.0; //CRIA A VARIÁVEL, INDICANDO QUE É CONSTANTE
```

Funções Matemáticas

Algumas das funções disponíveis da biblioteca **math.h** são:

Função	Finalidade
<code>abs (i)</code>	Retorna o valor absoluto de i
<code>ceil (d)</code>	Arredonda para cima, para o próximo valor inteiro maior que d
<code>cos (d)</code>	Retorna o cosseno de d
<code>floor (d)</code>	Arredonda para baixo, para o próximo valor inteiro menor que d
<code>log (d)</code>	Calcula o logaritmo neperiano $\log(d)$
<code>pow (d1, d2)</code>	Retorna d1 elevado a d2
<code>rand ()</code>	Retorna um inteiro positivo aleatório
<code>sin (d)</code>	Retorna o seno de d
<code>sqrt (d)</code>	Retorna a raiz quadrada de d
<code>tan (d)</code>	Retorna a tangente de d

As funções que possuem **retorno**, necessitam de uma variável para receber o valor retornado. Exemplo: **potencia = pow(10,2)**

Estrutura Sequencial em C

- **Bibliotecas** são arquivos contendo várias funções que podem ser incorporadas aos programas escritos em C.
- A diretiva `#include` faz com que o texto da biblioteca especificada seja inserido no programa.
- Exemplo:
 - A biblioteca `stdio.h` permite a utilização de diversos comandos de entrada e saída.

Estrutura inicial de código em C

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

```
int main() {
```

```
    // código fica aqui
```

```
}
```

Exercícios

1. Analise o seguinte trecho de código em C. Sua tarefa é identificar e corrigir os erros.

```
#include <stdio.h>

int main() {
    int idade = "20";
    float altura = 1.75;
    printf("Idade: %d\n", idade);
    printf("Altura: %.2f\n", altura);
    return 0;
}
```

Exercícios

2. O código a seguir declara duas variáveis, mas não as inicializa. Sua tarefa é modificar o código para atribuir valores a elas antes da impressão.

```
#include <stdio.h>

int main() {
    char letra;
    float preco;
    printf("Letra: %c\n", letra);
    printf("Preço: %.2f\n", preco);
    return 0;
}
```


Exercícios

3. Faça um programa que receba um número inteiro e imprima na tela o seu dobro.
4. Faça programa em C que receba as 3 notas de uma aluno. O programa deve exibir a nota final do aluno, que é calculada como a média das três notas.
5. Faça um programa que receba um número inteiro e escreva na tela o seu quadrado. Obs: use `pow(variável, 2);`
6. Faça um programa que receba a idade de uma pessoa e calcule quantos dias ela viveu.

Dúvidas?