Tipos de Dados, Variáveis e comandos de entrada e saída

DCC 119 – Algoritmos

Introdução

- Nesta aula, serão construídos programas de computador muito simples.
- Por enquanto, vamos assumir que todo programa tem a seguinte estrutura básica:

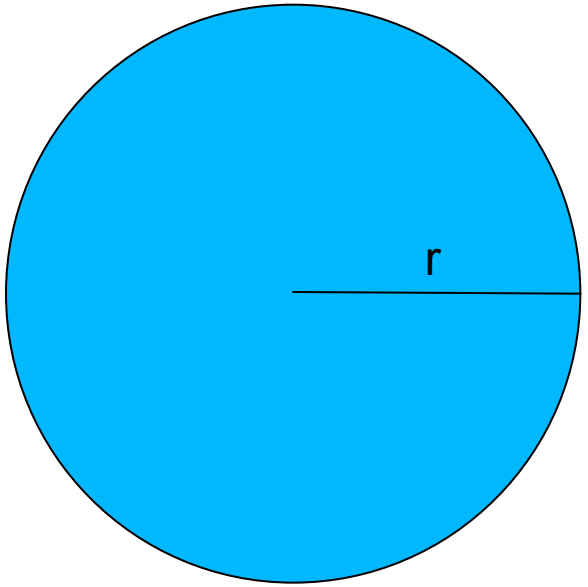
```
int main()
{
   ...
   return 0;
}
```

(quando estudarmos funções, os conceitos que definem esta estrutura serão vistos)

Introdução

- Um programa de computador utiliza diversos dados durante seu processamento.
- Exemplo:

Imagine um programa que calcule a área de um círculo.

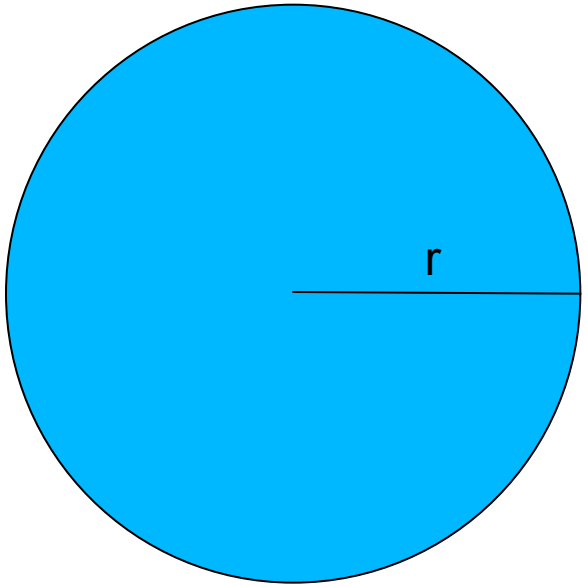


$$\text{Área} = \pi . r^2$$

Introdução

Algoritmo para calcular a área de um círculo:

1. Peça ao usuário que informe o comprimento do `raio`
2. Obtenha o valor da `área`, calculando `raio * raio * 3.14159`
3. Informe ao usuário o valor da `área`



Introdução

Quais valores numéricos aparecem no algoritmo?

Raio (número real): representa a medida do raio do círculo e seu valor pode variar dependendo do tamanho do círculo.

Pi (número real): representa a constante numérica 3,14159... Apresenta sempre o mesmo valor, independente do círculo.

Área (número real): representa a área de um círculo. Seu valor pode variar dependendo do tamanho do círculo.

Introdução

Assim como neste exemplo, um valor, em um programa, pode ser classificado como:

- **Constante**: dado cujo valor se manterá inalterado toda vez que o programa for utilizado.
- **Variável**: dado cujo valor pode ser modificado a cada execução ou, até mesmo, durante a execução do programa.

Constantes

- Uma constante pode ser representada no texto diretamente pelo seu valor.

```
int main()
{
   int a;
   float b;
   float c;
   a = 0;
   b = 2.5;
   c = 8.7 * b;
   return 0;
}
```

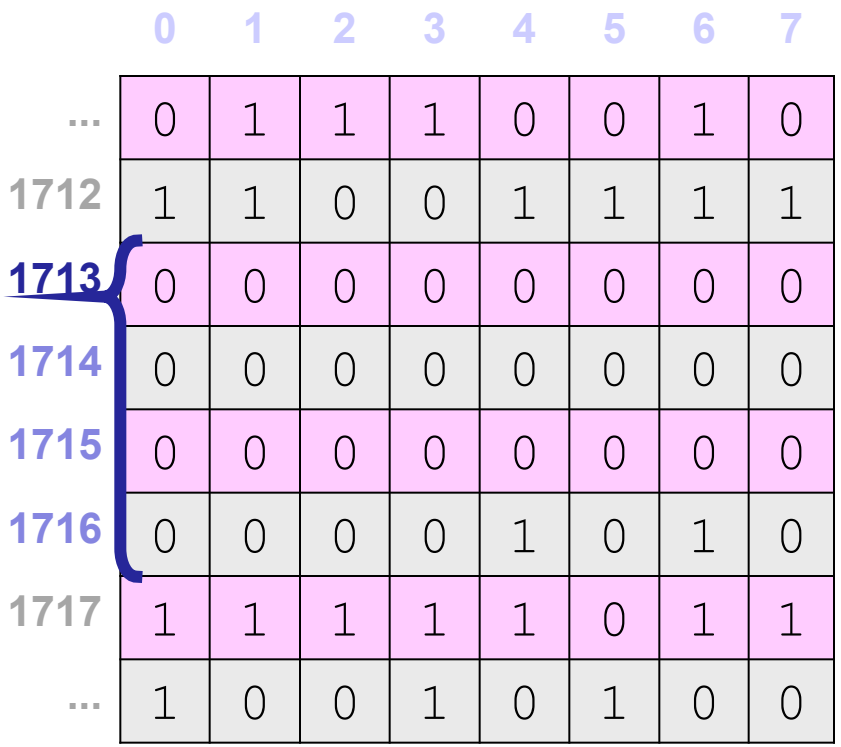
Variáveis

- Uma **variável** armazena um **valor** de determinado **tipo** que pode variar ao longo da execução do programa.
- Para cada variável, é reservado um espaço na memória do computador para armazenar seu valor.

Variáveis

Exemplo: para armazenar um número inteiro, o programa normalmente reserva 4 bytes de memória.

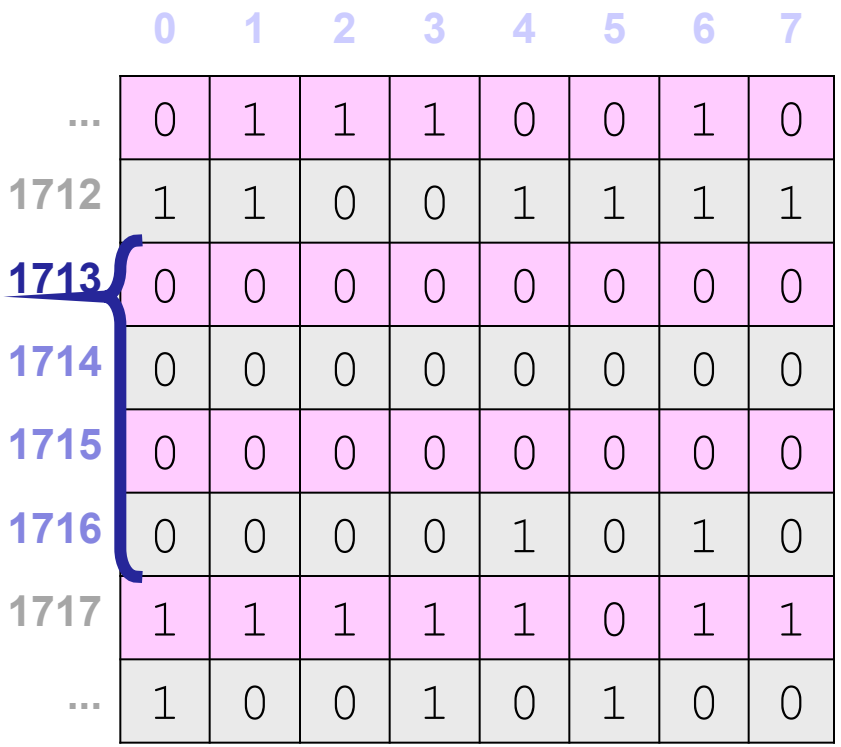
Observe que a memória do computador armaze-na apenas valores biná-rios – isto é, sequências compostas por 0's e 1's.



Variáveis

Exemplo: para armazenar um número inteiro, o programa normalmente reserva 4 bytes de memória.

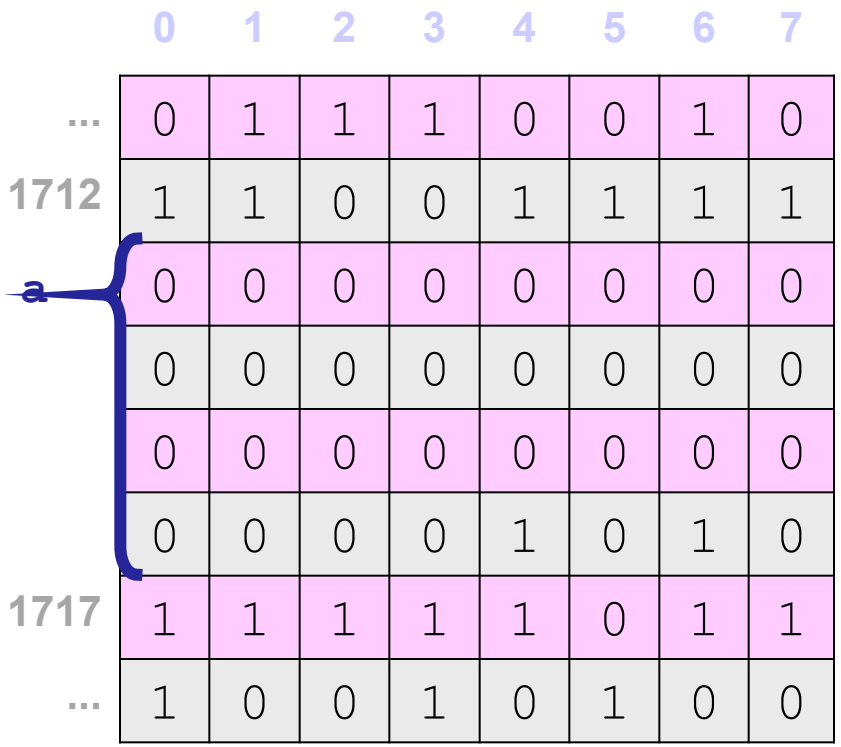
O número binário armazenado nestes 4 bytes representa o valor da variável (neste caso, 10).



Variáveis

- No texto de um programa, uma variável é representada por um identificador único.

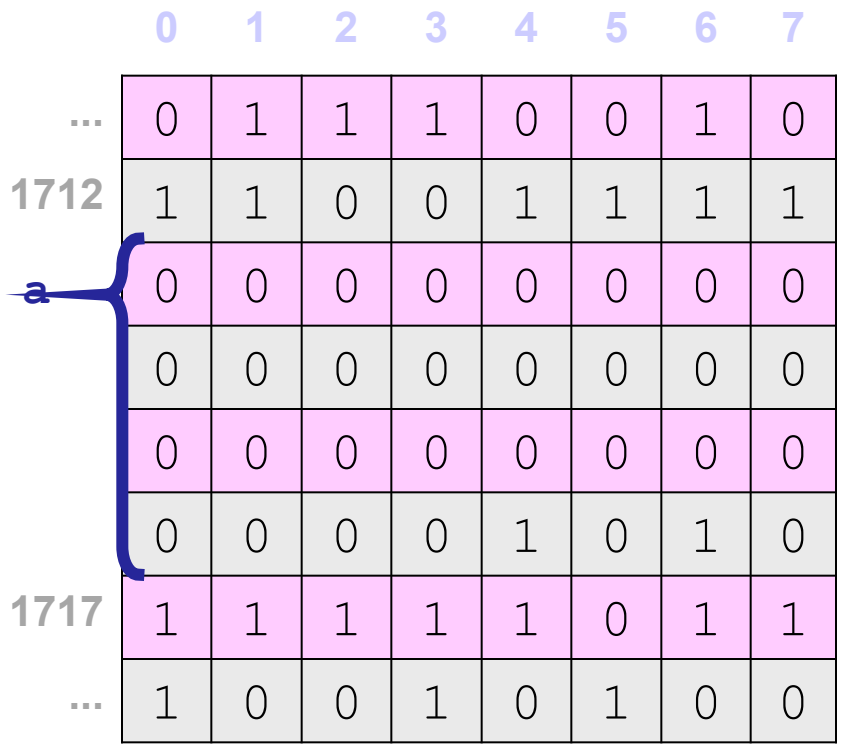
```
int main()
{
   int a;
   float b;
   float c;
   a = 0;
   b = 2.5;
   c = 8.7 * b;
   return 0;
}
```



Variáveis

- O valor da variável pode ser alterado ao longo do programa, mas seu nome permanece o mesmo.

```
int main()
{
   int a;
   float b;
   float c;
   a = 0;
   b = 2.5;
   c = 8.7 * b;
   return 0;
}
```



Variáveis

Identificador da variável:

- Nome único criado pelo programador.
- Não pode ser uma palavra reservada da linguagem C.

  Exemplos: `int`, `main`, `return`,...

- Pode conter apenas **letras**, **dígitos** e **sublinha.**
- Deve começar com uma **letra** (por padrão utilizam-se letras minúsculas).
- Deve permitir a identificação do valor que representa (ex: `raio`, `area`, etc).

Variáveis

Exemplos

Válidos:

```
nome
x1
nota_01
telefone
salario_base
nota2aProva
```

Inválidos:

```
1ano
salário
valor-1
endereço
salario/hora
2aProva
```

Tipos de dados

- Toda constante e toda variável de um programa tem um tipo de dados associado.
- Toda linguagem de programação contém um conjunto de tipos de dados pré-definido chamados de implícitos, primitivos ou básicos.

Tipos de dados básicos em C

`int` → utilizado para representar um número inteiro. Exemplo: `1`, `-5`, `1024` ,etc.

`float` ou **`double`** → utilizados para representar números reais (ponto flutuante). Exemplo: `-1.0`, `3.14159`, `2.718281`

`char` → utilizado para representar um único caractere (letra, dígito, símbolo, ponto, etc). Exemplo: `'a'`, `'A'`, `'5'`, `'@'`, `'!'`, etc.

Tipos de dados básicos em C

Atenção

- Para valores dos tipos **`float`** ou **`double`**, o separador decimal é o ponto.

```
3.14159
```


- Constantes do tipo de dados **`char`** sempre aparece entre aspas simples.

```
'a'
```

Tipos de dados

`booleano` → utilizado para representar um valor **lógico** que pode ser verdadeiro ou falso.

- Não é um tipo básico de C
- Em C, são representados por inteiros:
  - Falso: 0
  - Verdadeiro: 1 (ou qualquer valor diferente de 0)
- Várias operações da linguagem utilizam este tipo de dado, como veremos posteriormente.

Exercício

1) Indique quais das constantes abaixo são do tipo **`int`**:

( )1000 ( ) '8' ( )“-900”

( )-456 ( ) 34 ( )-1.56

2) Indique quais das constantes abaixo são do tipo **`float`**:

( ) -678.0 ( ) “0.87” ( ) “-9.12”

( ) -456.0 ( )“Cinco” ( ) -1.56

3) Indique quais das constantes abaixo são do tipo **`char`**:

( ) 'z' ( ) “onze” ( ) d

( ) 45 ( ) '8' ( ) 'F'

Declaração de variáveis

- A declaração de uma variável é o momento em que esta é criada no programa.

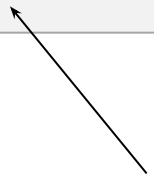
- Para criar uma variável, é necessário indicar:
  - o *tipo* da variável
  - o *identificador* da variável

```
int idade;
float peso, altura;
char sexo;
```

Declaração de variáveis

- Como todo comando simples em C, a declaração termina com um ponto e vírgula.

```
int idade;
float peso, altura;
char sexo;
```



Declaração de variáveis

Implicações da declaração de variáveis:

- É alocado um espaço na memória onde seja possível armazenar valores do tipo especificado.

Tipo	Espaço que ocupa na memória	Faixa
`char`	**1 byte**	*-128 a 127* (incluindo letras e símbolos)
`int`	**4 bytes***	-2147483648 a 2147483647*
`float`	**4 bytes**	*3.4E-38* a 3.4E+38 (6 casas de precisão)
`double`	**8 bytes**	*1.7E-308* a *1.7E+308* (15 casas de precisão)

* Estes valores podem variar dependendo da configuração da máquina.

Declaração de variáveis

Implicações da declaração de variáveis:

- O nome da variável é associado ao endereço de memória reservado.
  Assim, toda vez que a variável for referenciada, o computador vai trabalhar com o conteúdo de seu endereço de memória.

Declaração de variáveis

Observações importantes:

- Durante todo o programa, a variável armazenará apenas valores do tipo especificado na sua declaração.
- Uma variável só pode ser utilizada em um programa após sua declaração. Por isso, as declarações de variáveis são realizadas no início do programa.

Exercício

Indique as opções com declarações válidas:

```
(  ) char endereço;
(  ) int valor1, valor2;
(  ) float área;
(  ) int 21;
(  ) char a, b, char;
(  ) int a,
(  ) int a,b,a;
(  ) float f1,f2,f3,4f;
(  ) int meu_nro;
(  ) float leitura_sensor;
```

Operador de atribuição

Um comando de atribuição é a forma mais básica de modificar o valor de uma variável:

```
nomeVar = expressao;
```

`nomeVar`

- identificador da variável que será modificada
- apenas **uma** variável pode ser modificada por vez
- o nome da variável fica **sempre no lado esquerdo** do operador de atribuição

Operador de atribuição

Um comando de atribuição é a forma mais básica de modificar o valor de uma variável:

```
nomeVar = expressao;
```

`operador =`

- operador de atribuição
- para não confundir com o operador de comparação, evite ler **`var=10;`** como "*var é igual a 10*"; normal-mente, lê-se "*var* ***recebe*** *10*"

Operador de atribuição

Um comando de atribuição é a forma mais básica de modificar o valor de uma variável:

```
nomeVar = expressao;
```

`expressao`

- expressão cujo resultado será armazenado na variável
- pode ser composta por um valor constante, uma outra variável ou uma expressão (matemática ou lógica) que utilize constantes e variáveis, etc
- fica **sempre do lado direito** do operador de atribuição

Operador de atribuição

Um comando de atribuição é a forma mais básica de modificar o valor de uma variável:

```
nomeVar = expressao;
```

`;` (ponto e vírgula)

- como todo comando simples em C, o comando de atribuição é finalizado com um ponto e vírgula.

Operador de atribuição

Exemplos:

```
raio = 2.5;
area = 3.14159 * (raio * raio);
raio2 = raio;
sexo = 'F';
delta = (b * b) - 4 * a * c;
digito = '5';
```

Inicialização de variáveis

- Quando uma variável é declarada, seu valor inicial não é modificado e seu conteúdo não é conhecido. Por isso, dizemos que a variável contem ***lixo***.

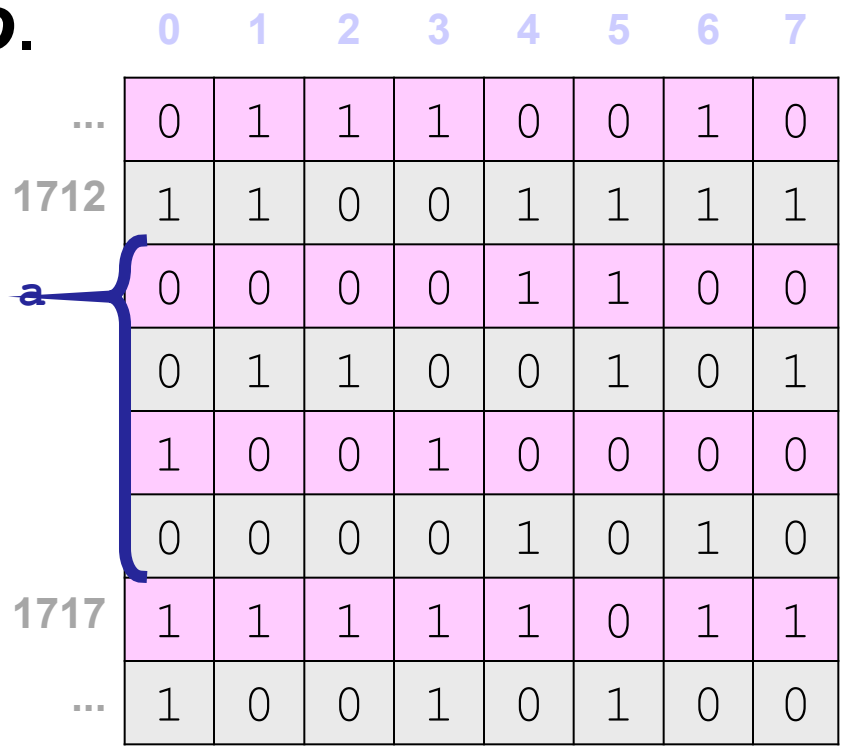
```
int main()
{
   int a;
   int b;
   b = a;
   a = 10;
   return 0;
}
```

	0	1	2	3	4	5	6	7
...	0	1	1	1	0	0	1	0
1712	1	1	0	0	1	1	1	1
1713	0	0	0	0	1	1	0	0
1714	0	1	1	0	0	1	0	1
1715	1	0	0	1	0	0	0	0
1716	0	0	0	0	1	0	1	0
1717	1	1	1	1	1	0	1	1
...	1	0	0	1	0	1	0	0

Inicialização de variáveis

- Quando uma variável é declarada, seu valor inicial não é modificado e seu conteúdo não é conhecido. Por isso, dizemos que a variável contem ***lixo***.

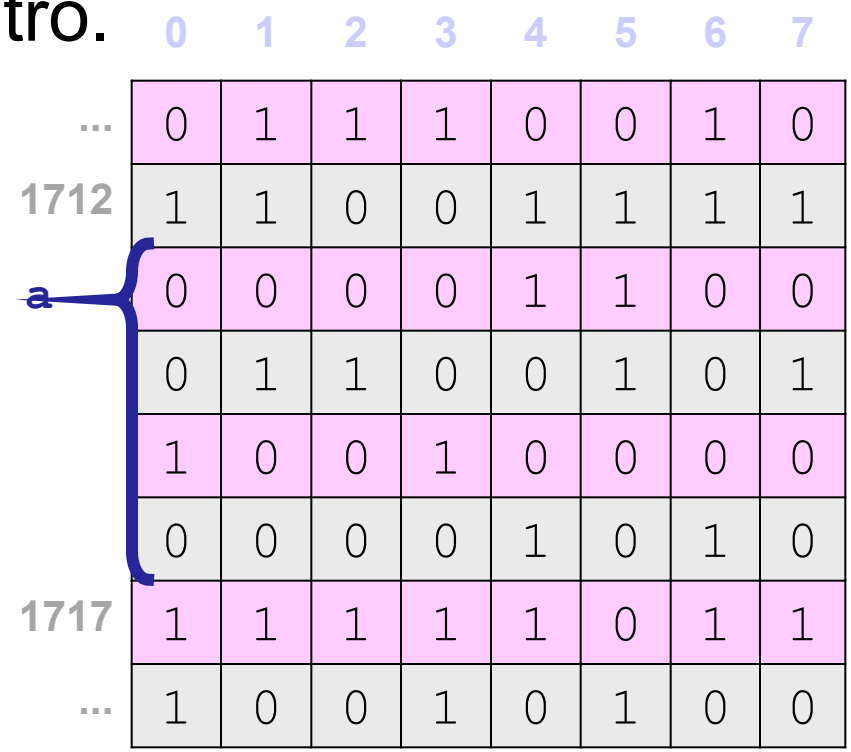
```
int main()
{
   int a;
   int b;
   b = a;
   a = 10;
   return 0;
}
```



Inicialização de variáveis

- Nenhuma variável deve ser utilizada antes de ser inicializada! No exemplo abaixo, b recebe 11000110010110010000000001010 = 207982602
  Em outra execução do mesmo programa, o valor provavelmente será outro.

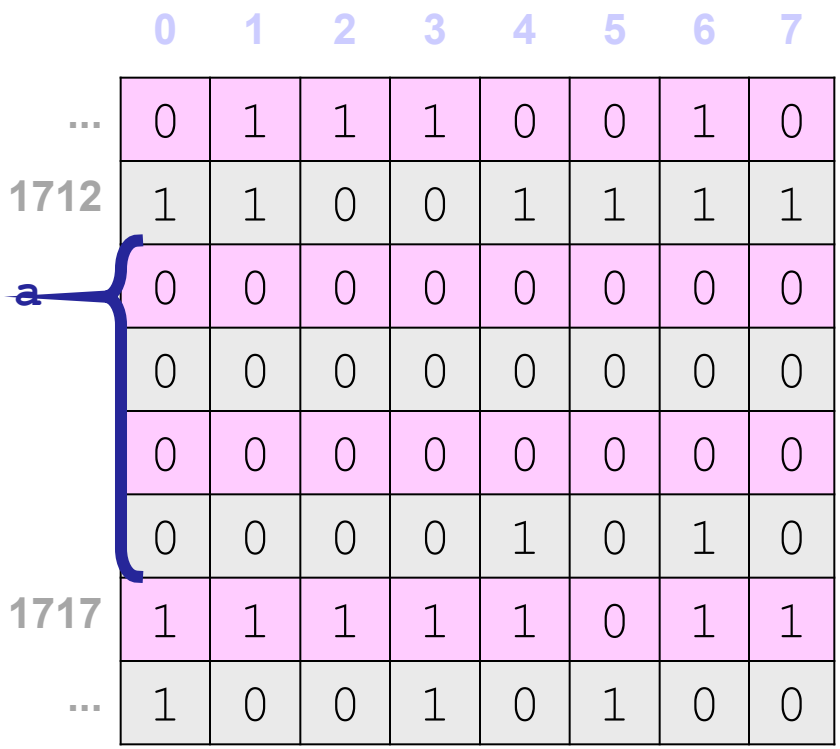
```
int main()
{
   int a;
   int b;
   b = a;
   a = 10;
   return 0;
}
```



Inicialização de variáveis

- A inicialização da variável deve ser realizada antes que seu valor apareça em uma expressão ou comando do programa.

```
int main()
{
   int a;
   int b;
   a = 10;
   b = a;
   return 0;
}
```



Inicialização de variáveis

- Em C, é possível inicializar uma variável em sua declaração:

```
int a = 10;
char setor = '1';
float elem1 = 0.0, elem2 = 1.0;
int x = 5, y, z = 20;
```

Obs: no exemplo acima, apenas a variável `y` não foi inicializada.

Inicialização de variáveis

Atenção

esquecer de inicializar o valor de uma variável é um erro lógico comum e difícil de detectar...

- Como acontece: variável é declarada sem que seu valor seja explicitamente inicializado.
- Problema: valor da variável é desconhecido (“lixo”)

```
// Erro !!!
// Variável y não foi
// inicializada e
// contém “lixo” de
// memória

float x, y, z;
x = 1.0;
z = x + y;
```

Expressões

Combinação de variáveis, constantes e operadores que, quando avaliada, resulta em um valor.

Expressão aritmética:
resulta em um número (inteiro ou real).

Expressão lógica:
resulta em VERDADEIRO ou FALSO.

Expressões aritméticas

```
10
a + 15
base * altura
90 / 4.0
3.065
189 % divisor
(x1 - 5) * x2
```

Variáveis:

a, base, altura, divisor, x1, x2

Constantes:

10, 15, 90, 4.0, 3.065, 189, 5

Operadores:

+, *, /, %, -

Expressões aritméticas

Operadores

			INTEIROS	REAIS
UNÁRIO	-	sinal negativo	-2 -a	-2.0 -b
BINÁRIOS	+	adição	a + 2	b + 2.0
	-	subtração	a – 2	b – 2.0
	*	multiplicação	a * 2	b * 2.0
	/	divisão	a / 2	b / 2.0
	%	módulo (resto da divisão)	a % 2	(operação não definida para reais)

Expressões aritméticas

Operadores → Para a = 5 e b = 5.0:

			INTEIROS	REAIS
UNÁRIO	-	sinal negativo	-2 -a → -5	-2.0 -b → -5.0
BINÁRIOS	+	adição	a + 2 → 7	b + 2.0 → 7.0
	-	subtração	a – 2 → 3	b – 2.0 → 3.0
	*	multiplicação	a * 2 → 10	b * 2.0 → 10.0
	/	divisão	a / 2 → 2	b / 2.0 → 2.5
	%	módulo (resto da divisão)	a % 2 → 1	(operação não definida para reais)

Expressões aritméticas

Operações aritméticas mais complexas:

```
pow(base,2)
sqrt(16)
sin(x)
cos(x)
cos(2*x)
sin(x)*cos(y)
abs(-5)
```

Veremos mais detalhes sobre funções adiante...

Expressões lógicas

Envolvem os operadores:

Relacionais:
igual (==), diferente (!=),
menor que (<), menor ou igual a (<=),
maior que (>), maior ou igual a (>=)

Lógicos:
negação (!), conjunção (&&), disjunção (||)

Expressões lógicas

- Sempre resultam em VERDADEIRO ou FALSO.

- Serão abordadas mais detalhadamente na introdução de **estruturas condicionais**.

Avaliação de expressões

Prioridade para execução de operações em uma expressão:

1. Parênteses (dos mais internos para os mais externos);
2. Expressões aritméticas, seguindo a ordem: funções, multiplicação e divisão, adição e subtração;
3. Expressões lógicas relacionais: <, <=, ==, >, >= e !=;
4. Expressões lógicas, seguindo a ordem: negação, conjunção, disjunção;
5. Da esquerda para a direita, quando houver indeterminações.

Na dúvida, use parênteses em suas expressões.

Exercício

Dadas as declarações:

```
int a, b;
float x, y;
```

Indique as opções com expressões válidas:

```
(  ) a + b = 2;
(  ) a = a % b;
(  ) x = y + a;
(  ) y = x % y;
(  ) 2 = a + b - a / 3;
(  ) b = (a + b - a) / 3;
(  ) b = (a + b) - (a / 3);
(  ) x = sqrt(y) * 7;
(  ) x * x * x = pow(x,3);
(  ) y = x * x * x - pow(x,3);
```

Exercício

1. Construa uma sequência de instruções para calcular o volume de um copo com 12 cm de altura e 6 cm de diâmetro, da seguinte forma:

- Declare as variáveis para raio, altura e volume;
- Inicialize as variáveis cujo valor é conhecido;
- Calcule o valor do volume e armazene-o na variável.

2. Construa uma sequência de instruções para indicar quantos dias, horas, minutos e segundos equivalem a 200.000 segundos. Assim como no exercício anterior, declare variáveis, inicialize-as e, por fim, realize o cálculo armazenando o resultado.

Impressão na tela

A função **`printf`** escreve um texto no dispositivo de saída padrão do computador (isto é, na tela do computador).

```
int main()
{
   printf("Alo mundo!");
   return 0;
}
```

Impressão na tela

A função **`printf`** escreve um texto no dispositivo de saída padrão do computador (isto é, na tela do computador).

```c
#include <stdio.h>

int main()
{
  printf("Estou aprendendo a programar em C");
  return 0;
}
```

Impressão na tela

- Nos exemplos já vistos, a função **`printf`** escreve na tela apenas textos constantes.
- **`printf`** também pode ser utilizada para escrever valores armazenados em variáveis (por exemplo: resultados de cálculos realizados no programa).
- A sintaxe da função **`printf`** permite que seja possível imprimir, em um único comando:
  - apenas um texto (sem valores variáveis); ou
  - uma ou mais variáveis de diferentes tipos.

Sintaxe da função `printf`

```
printf( "Expressão" , Lista de argumentos );
```

A Expressão pode conter:

- *mensagens a serem* exibidas;
- *códigos de formatação* que indicam como o conteúdo de uma variável deve ser exibido; e
- *códigos especiais* para a exibição de alguns caracteres especiais.

Sintaxe da função `printf`

```
printf( "Expressão" , Lista de argumentos );
```

A Lista de argumentos (opcional) indica que valores deverão ser impressos. Podem ser:

- identificadores de variáveis,
- expressões aritméticas,
- valores constantes.

Estes valores devem aparecer numa lista separada por vírgulas.

Sintaxe da função `printf`

```
printf("Imprime o inteiro %d.",
       valorInteiro);
```

O código de formatação `%d` na expressão, indica que um valor inteiro deverá ser impresso na posição do texto onde o `%d` se encontra.

Sintaxe da função `printf`

```
int a = 10;
printf("\n Imprime variavel inteira: %d",a);
printf("\n Imprime constante inteira: %d",34);
printf("\n Imprime resultado: %d", (a*2)+5);
printf("\n Imprime expressao 1: (a*%d)+%d=%d",
        2, 5,(a*2)+5);
printf("\n Imprime expressao 2: (%d*%d)+%d=%d",
        a, 2, 5,(a*2)+5);
```

```
Imprime variavel inteira: 10
Imprime constante inteira: 34
Imprime resultado: 25
Imprime expressao 1: (a*2)+5=25
Imprime expressao 2: (10*2)+5=25
```

Sintaxe da função `printf`

```
printf("Imprime o valor real %f.",
                valorReal);
```

O código de formatação `%f` na expressão, indica que um valor flutuante (float ou double) deverá ser impresso na posição do texto onde o `%f` se encontra.

Sintaxe da função `printf`

```
float x = 1.0 / 3;
printf("\n Imprime variavel real: %f",x);
printf("\n Imprime constante real: %f",8.4);
printf("\n Imprime resultado: %f", 5.0/2 );
printf("\n Imprime expressao 1: (x*%d)+%d=%f",
        2, 5,(x*2)+5);
printf("\n Imprime expressao 2: (%f*%d)+%d=%f",
        x, 2, 5,(x*2)+5);
```

```
Imprime variavel real: 0.333333
Imprime constante real: 8.400000
Imprime resultado: 2.500000
Imprime expressao 1: (x*2)+5=5.666666
Imprime expressao 2: (0.333333*2)+5=5.666666
```

Impressão de Tipos de Dados

Código	Tipo	Elemento armazenado
`%c`	`char`	um único caractere
`%d`	`int`	um inteiro
`%f`	`float`	um número em ponto flutuante
`%lf`	`double`	ponto flutuante com dupla precisão
`%e`	`float` ou `double`	um número na notação científica
`%s`	(tipo composto)	uma cadeia de caracteres

Sintaxe da função `printf`

```
char var = 'B';
printf("\n Imprime variavel char: %c",var);
printf("\n Imprime constante char: %c",'Z');
printf("\n Imprime sequencia de caracteres: %s",
            "qualquer \ntexto\n!!!");
```

```
Imprime variavel char: B
Imprime constante char: Z
Imprime sequencia de caracteres: qualquer
texto
!!!
```

Exercício

3. Elabore um programa completo que imprima o dobro, o triplo e o quadrado do valor x. O valor de x pode ser escolhido por você ao inicializar a variável. Supondo que o valor de x é 3, a saída de seu programa deve ser:

```
Valor: 3
Dobro de 3: 6
Triplo de 3: 9
Quadrado de 3: 9
```

Use variáveis para armazenar os valores numéricos que deverão ser impressos.

---

4. Execute as instruções ao lado, indicando o que será impresso pelo programa.

```
01 int main() {
02   float a=1.0, b=-2.0, c=1.0;
03   float d, r1, r2;
04   d=b*b-4*a*c;
05   r1=(-b+sqrt(d))/(2*a);
06   r2=(-b-sqrt(d))/(2*a);
07   printf("%f e %f",r1,r2);
08   return 0;
09 }
```

Leitura de dados do teclado

A função **`scanf`** armazena em uma variável dados informados pelo usuário através do teclado.

```
int main()
{
   float raio, area;
   printf("Digite o raio do circulo: ");
   scanf("%f", &raio );
   area = 3.14159 * raio * raio;
   printf("Area do circulo: %f", area);
   return 0;
}
```

Sintaxe da função `scanf`

```
scanf( "Codigos de Formatacao" ,
Lista de endereços de variáveis );
```

Os Códigos de Formatação indicam o tipo e a ordem esperada dos valores que serão lidos.

A Lista de endereços de variáveis contém uma sequência de variáveis dos tipos indicados nos códigos de formatação, todas precedidas por &.

Cada expressão &NomeDaVariavel retorna o endereço da respectiva variável na memória. Isto permite que o programa saiba onde deve ser armazenado o valor lido do teclado.

Sintaxe da função `scanf`

```
scanf( "Codigos de Formatacao" ,
Lista de endereços de variáveis );
```

Na função **`scanf`**:

- A Lista de endereços de variáveis NÃO é opcional.
- A Lista de endereços de variáveis NÃO aceita valores constantes ou expressões.

Sintaxe da função `scanf`

```
int a,b;
float x,y;
printf("\n Digite valor inteiro: ");
scanf("%d",&a);
printf("\n Digite valor real: ");
scanf("%f",&x);
printf("\n Digite valores inteiro e real");
scanf("%d%f",&b,&y);
printf("\n a=%d x=%f b=%d y=%f",a,x,b,y);
```

```
Digite valor inteiro: 25
Digite valor real: 6.158
Digite valores inteiro e real: 3  9.8
a=25 x=6.158000 b=3 y=9.800000
```

Leitura de dados do teclado

- A leitura de dados utilizando **`scanf`** é uma forma alternativa de inicialização de variáveis.

```
int main()
{
  float raio, area;
  printf("Digite o raio do circulo: ");
  scanf("%f", &raio );
  area = 3.14159 * raio * raio;
  printf("Area do circulo: %f", area);
  return 0;
}
```

Leitura de dados do teclado

```c
#include <stdio.h>

int main()
{
   int num;
   printf("Digite um valor: ");
   scanf("%d",&num);
   printf("\nO valor digitado foi %d.",num);
   return 0;
}
```

Leitura de dados do teclado

```c
#include <stdio.h>

int main()
{
   int n1,n2,soma;
   printf("Digite dois valores: ");
   scanf("%d",&n1);
   scanf("%d",&n2);
   soma = n1 + n2;
   printf("\n%d + %d = %d.",n1,n2,soma);
   return 0;
}
```

Leitura de dados do teclado

```c
#include <stdio.h>

int main()
{
   int idade;
   float altura;
   printf("Digite sua idade e sua altura: ");
   scanf("%d%f", &idade, &altura);
   ...
```

Formatação de dados

- **`scanf`** e **`printf`** têm várias outras opções para formatação de dados.

- Estas opções serão vistas em maiores detalhes na aula de Laboratório de Programação.

Teste de Mesa

- Técnica utilizada pelo programador para verificar se o algoritmo (ou programa) está correto, isto é, se alcança o resultado esperado.
- Um teste de mesa simula a execução de um algoritmo utilizando apenas papel e caneta, sem de fato executá-lo no computador.

Teste de Mesa

Durante o teste de mesa, para simular a execução de um algoritmo, o programador deve observar:

- a sequência (a ordem) em que as linhas do código são executadas;
- o valor de cada uma das variáveis do algoritmo durante toda a execução;
- a saída produzida pelo programa, isto é, as impressões realizadas ao longo da execução.

Teste de Mesa

Assim, para simular uma execução, você deve:

1. Numerar as linhas de código;
2. Manter uma tabela com o número da linha e o valor de cada variável;
3. Armazenar o conteúdo impresso.

TESTE DE MESA

linha	var1	var2	var3	...

Teste de Mesa

```
 1  #include <stdio.h>
 2
 3  /* Programa que calcula a área de um círculo */
 4  int main()
 5  {
 6    //Declara variáveis
 7    float raio, area;
 8    //Imprime informações para o usuário
 9    printf(" Programa que calcula area de ");
10    printf(" um circulo.\n Digite o raio: ");
11    //Lê dado de entrada
12    scanf("%f", &raio );
13    //Calcula area
14    area = 3.14159 * raio * raio;
15    //Imprime resultado
16    printf(" Area do circulo: %f", area);
17    return 0;
18  }
```

TESTE DE MESA

linha	raio	area

Teste de Mesa

A execução começa em **`main()`**.
As variáveis começam com lixo.

```
 1  #include <stdio.h>
 2
 3  /* Programa que calcula a área de um círculo */
 4  int main()
 5  {
 6    //Declara variáveis
 7    float raio, area;
 8    //Imprime informações para o usuário
 9    printf(" Programa que calcula area de ");
10    printf(" um circulo.\n Digite o raio: ");
11    //Lê dado de entrada
12    scanf("%f", &raio );
13    //Calcula area
14    area = 3.14159 * raio * raio;
15    //Imprime resultado
16    printf(" Area do circulo: %f", area);
17    return 0;
18  }
```

TESTE DE MESA

linha	raio	area
4	?	?

Teste de Mesa

Na execução, linhas que não possuem instrução são puladas.

```
 1  #include <stdio.h>
 2
 3  /* Programa que calcula a área de um círculo */
 4  int main()
 5  {
 6    //Declara variáveis
 7    float raio, area;
 8    //Imprime informações para o usuário
 9    printf(" Programa que calcula area de ");
10    printf(" um circulo.\n Digite o raio: ");
11    //Lê dado de entrada
12    scanf("%f", &raio );
13    //Calcula area
14    area = 3.14159 * raio * raio;
15    //Imprime resultado
16    printf(" Area do circulo: %f", area);
17    return 0;
18  }
```

TESTE DE MESA

linha	raio	area
4	?	?
7	?	?

Teste de Mesa

```
1  #include <stdio.h>
2
3  /* Programa que calcula a área de um círculo */
4  int main()
5  {
6    //Declara variáveis
7    float raio, area;
8    //Imprime informações para o usuário
9    printf(" Programa que calcula area de ");
10   printf(" um circulo.\n Digite o raio: ");
11   //Lê dado de entrada
12   scanf("%f", &raio );
13   //Calcula area
14   area = 3.14159 * raio * raio;
15   //Imprime resultado
16   printf(" Area do circulo: %f", area);
17   return 0;
18 }
```

TESTE DE MESA

linha	raio	area
4	?	?
7	?	?
9	?	?

Programa que calcula area de

Teste de Mesa

```
1  #include <stdio.h>
2
3  /* Programa que calcula a área de um círculo */
4  int main()
5  {
6    //Declara variáveis
7    float raio, area;
8    //Imprime informações para o usuário
9    printf(" Programa que calcula area de ");
10   printf(" um circulo.\n Digite o raio: ");
11   //Lê dado de entrada
12   scanf("%f", &raio );
13   //Calcula area
14   area = 3.14159 * raio * raio;
15   //Imprime resultado
16   printf(" Area do circulo: %f", area);
17   return 0;
18 }
```

TESTE DE MESA

linha	raio	area
4	?	?
7	?	?
9	?	?
10	?	?

```
Programa que calcula area de  um circulo.
Digite o raio:
```

Teste de Mesa

Vamos assumir que o usuário forneceu como entrada o valor **1**.

```
 1  #include <stdio.h>
 2
 3  /* Programa que calcula a área de um círculo */
 4  int main()
 5  {
 6    //Declara variáveis
 7    float raio, area;
 8    //Imprime informações para o usuário
 9    printf(" Programa que calcula area de ");
10    printf(" um circulo.\n Digite o raio: ");
11    //Lê dado de entrada
12    scanf("%f", &raio );
13    //Calcula area
14    area = 3.14159 * raio * raio;
15    //Imprime resultado
16    printf(" Area do circulo: %f", area);
17    return 0;
18  }
```

TESTE DE MESA

linha	raio	area
4	?	?
7	?	?
9	?	?
10	?	?
12	1.0	?

```
Programa que calcula area de  um circulo.
Digite o raio: 1
```

Teste de Mesa

O cálculo é realizado usando o valor armazenado em **raio**.

```
 1  #include <stdio.h>
 2
 3  /* Programa que calcula a área de um círculo */
 4  int main()
 5  {
 6    //Declara variáveis
 7    float raio, area;
 8    //Imprime informações para o usuário
 9    printf(" Programa que calcula area de ");
10    printf(" um circulo.\n Digite o raio: ");
11    //Lê dado de entrada
12    scanf("%f", &raio );
13    //Calcula area
14    area = 3.14159 * raio * raio;
15    //Imprime resultado
16    printf(" Area do circulo: %f", area);
17    return 0;
18  }
```

TESTE DE MESA

linha	raio	area
4	?	?
7	?	?
9	?	?
10	?	?
12	1.0	?
14	1.0	3.14159

```
Programa que calcula area de  um circulo.
Digite o raio: 1
```

Teste de Mesa

O programa imprime o valor armazenado em **area**.

```
 1  #include <stdio.h>
 2
 3  /* Programa que calcula a área de um círculo */
 4  int main()
 5  {
 6    //Declara variáveis
 7    float raio, area;
 8    //Imprime informações para o usuário
 9    printf(" Programa que calcula area de ");
10    printf(" um circulo.\n Digite o raio: ");
11    //Lê dado de entrada
12    scanf("%f", &raio );
13    //Calcula area
14    area = 3.14159 * raio * raio;
15    //Imprime resultado
16    printf(" Area do circulo: %f", area);
17    return 0;
18  }
```

TESTE DE MESA

linha	raio	area
4	?	?
7	?	?
9	?	?
10	?	?
12	1.0	?
14	1.0	3.14159
16	1.0	3.14159

```
Programa que calcula area de  um circulo.
Digite o raio: 1
Area do circulo: 3.141590
```

Teste de Mesa

O comando **return** encerra a execução do programa.

```
 1  #include <stdio.h>
 2
 3  /* Programa que calcula a área de um círculo */
 4  int main()
 5  {
 6    //Declara variáveis
 7    float raio, area;
 8    //Imprime informações para o usuário
 9    printf(" Programa que calcula area de ");
10    printf(" um circulo.\n Digite o raio: ");
11    //Lê dado de entrada
12    scanf("%f", &raio );
13    //Calcula area
14    area = 3.14159 * raio * raio;
15    //Imprime resultado
16    printf(" Area do circulo: %f", area);
17    return 0;
18  }
```

TESTE DE MESA

linha	raio	area
4	?	?
7	?	?
9	?	?
10	?	?
12	1.0	?
14	1.0	3.14159
16	1.0	3.14159
17	1.0	3.14159

```
Programa que calcula area de  um circulo.
Digite o raio: 1
Area do circulo: 3.141590
```

Exercícios

5. Faça um programa que lê uma temperatura em graus Celsius e apresenta-a convertida em graus Fahrenheit. A fórmula de conversão é:

$$F \leftarrow (9*C+160)/5$$

6. Faça um programa que lê um valor de salário mínimo e o salário de um funcionário. O programa deve calcular e imprimir quantos salários mínimos esse funcionário ganha.
   Após fazer o programa, verifique se seu programa está correto fazendo o teste de mesa com as entradas `800.00` e `2030.40`.

Exercícios

7. O que é impresso no programa ao lado? Observe a formatação e o tipo de dado.

```
01 int main() {
02   int a=5;
03   printf("   %d\n + ",a);
04   printf("%d\n------",a/2);
05   printf("\n   %d",3*a/2);
06   return 0;
07 }
```

8. Reescreva o programa ao lado usando apenas duas variáveis.

```
01 int main() {
02   float lado1,lado2,lado3;
03   float perimetro;
04   printf("TRIANGULO\n");
05   printf("Digite os lados:");
06   scanf("%f%f%f",&lado1,
07         &lado2,&lado3);
08   perimetro=lado1+lado2+lado3;
09   printf("Perimetro:%f",perimetro);
10   return 0;
11 }
```

Exercícios

9. Construa um programa que aplique um desconto de 25% sobre o preço de um produto recebido como entrada e imprima o valor resultante. Verifique se o programa está correto fazendo o teste de mesa. Use o valor `150.00` como entrada.

10. Construa um programa para ler do teclado um intervalo de tempo em segundos, converter para horas, minutos e segundos e imprimir o resultado. Faça o teste de mesa para uma entrada de `72000` segundos.

Exercícios

DESAFIO: O programa abaixo (embora pareça inútil para um programador iniciante) executa uma tarefa bastante comum em algoritmos mais avançados.

```
01 int main() {
02   int valor1,valor2,auxiliar;
03   printf("Digite os valores:");
04   scanf("%d%d",&valor1,&valor2);
05   auxiliar = valor1;
06   valor1 = valor2;
07   valor2 = auxiliar;
08   printf("Valor 1: %d\n",valor1);
09   printf("Valor 2: %d\n",valor2);
10   return 0;
11 }
```

Responda: é possível inverter o conteúdo das duas variáveis sem utilizar a variável auxiliar? Justifique.

Exercícios

DESAFIO: Uma empresa contratou um médico para avaliar todos os seus funcionários na própria sede da empresa. Para que cada funcionário saiba o horário agendado para sua consulta médica, você deverá fazer um programa que lê a matrícula do funcionário e informa o dia e horário da consulta. Observe que:

- As matrículas dos funcionários são números consecutivos entre 1 e 30 (inclusive). Os funcionários serão atendidos em ordem crescente de matrícula.
- As consultas duram uma hora e serão realizadas em uma única semana, de 2ª a 6ª. O médico estará disponível das 8 às 14h.

Para a matrícula 24, por exemplo, o programa deverá imprimir a saída: `5ª-feira as 13:00 horas`

Tipos de Dados, Variáveis e Entrada e Saída em C

DCC 120 – Laboratório de Programação

Introdução à Programação

Conteúdo desta semana:

- Variáveis
- Tipos de dados
- Operador de atribuição
- Expressões
- Impressão de dados na tela
- Leitura de dados do teclado

Introdução à Programação

```c
/* Programa que calcula
 * área de um círculo
 */
int main()
{
  //Declara variáveis
  float raio, area;

  //Imprime informações para o usuário
  printf(" Programa que calcula");
  printf(" area de um circulo.\n");
  printf(" Digite o raio: ");

  //Lê dado de entrada
  scanf("%f", &raio );

  //Calcula area
  area = 3.14159 * raio * raio;

  //Imprime resultado
  printf(" Area do circulo: %f", area);
  return 0;
}
```

Estrutura básica de programas

```c
/* Programa que calcula
 * área de um círculo
 */
int main()
{
  //Declara variáveis
  float raio, area;

  //Imprime informações para o usuário
  printf(" Programa que calcula");
  printf(" area de um circulo.\n");
  printf(" Digite o raio: ");

  //Lê dado de entrada
  scanf("%f", &raio );

  //Calcula area
  area = 3.14159 * raio * raio;

  //Imprime resultado
  printf(" Area do circulo: %f", area);
  return 0;
}
```

Constantes

```
/* Programa que calcula
 * área de um círculo
 */
int main()
{
  //Declara variáveis
  float raio, area;

  //Imprime informações para o usuário
  printf(" Programa que calcula");
  printf(" area de um circulo.\n");
  printf(" Digite o raio: ");

  //Lê dado de entrada
  scanf("%f", &raio );

  //Calcula area
  area = 3.14159 * raio * raio;

  //Imprime resultado
  printf(" Area do circulo: %f", area);
  return 0;
}
```

Variáveis

```
/* Programa que calcula
 * área de um círculo
 */
int main()
{
  //Declara variáveis
  float raio, area;

  //Imprime informações para o usuário
  printf(" Programa que calcula");
  printf(" area de um circulo.\n");
  printf(" Digite o raio: ");

  //Lê dado de entrada
  scanf("%f", &raio );

  //Calcula area
  area = 3.14159 * raio * raio;

  //Imprime resultado
  printf(" Area do circulo: %f", area);
  return 0;
}
```

- Armazenam valores em um endereço específico na memória.
- Armazenam apenas valores de seu tipo.
- Têm nome único composto apenas por letras, dígitos e sublinha.
- São criadas na declaração:

```
tipo identificador;
```

Tipos básicos de dados

```c
/* Programa que calcula
 * área de um círculo
 */
int main()
{
  //Declara variáveis
  float raio, area;

  //Imprime informações para o usuário
  printf(" Programa que calcula");
  printf(" area de um circulo.\n");
  printf(" Digite o raio: ");

  //Lê dado de entrada
  scanf("%f", &raio );

  //Calcula area
  area = 3.14159 * raio * raio;

  //Imprime resultado
  printf(" Area do circulo: %f", area);
  return 0;
}
```

- `int` – valores inteiros
- `float ou double` – valores reais
- `char` – caracteres como letras, dígitos e símbolos
- `booleano` – valores FALSO e VERDADEIRO (não existe em C, mas pode ser representado através de inteiros)

Operador de atribuição

```c
/* Programa que calcula
 * área de um círculo
 */
int main()
{
  //Declara variáveis
  float raio, area;

  //Imprime informações para o usuário
  printf(" Programa que calcula");
  printf(" area de um circulo.\n");
  printf(" Digite o raio: ");

  //Lê dado de entrada
  scanf("%f", &raio );

  //Calcula area
  area = 3.14159 * raio * raio;

  //Imprime resultado
  printf(" Area do circulo: %f", area);
  return 0;
}
```

- Armazena o resultado da expressão do lado direito na variável do lado esquerdo do operador.
- Lê-se:
  área recebe ...

Expressão

```
/* Programa que calcula
 * área de um círculo
 */
int main()
{
  //Declara variáveis
  float raio, area;

  //Imprime informações para o usuário
  printf(" Programa que calcula");
  printf(" area de um circulo.\n");
  printf(" Digite o raio: ");

  //Lê dado de entrada
  scanf("%f", &raio );

  //Calcula area
  area = 3.14159 * raio * raio;

  //Imprime resultado
  printf(" Area do circulo: %f", area);
  return 0;
}
```

- Valores negativos (-)
- Adição (+)
- Subtração (-)
- Multiplicação (*)
- Divisão inteira (/)
- Módulo – resto da divisão inteira (%)
- Divisão real – pelo menos um operando real (/)

Conversão de tipo

- Conversões automáticas de valores na avaliação de uma expressão.

```
int a, b;
float c, d;
a = 5.5;     //conversao implicita de real para int
b = a/2.0;   //divisão com resultado real e conversão
c = a/2;     //divisão com resultado inteiro e conversão
d = a/2.0;   //divisão com resultado real
printf("a=%d, b=%d, c=%f, d=%f",
        a, b, c, d);
```

```
a=5, b=2, c=2.000000, d=2.500000
```

Conversão de tipo - Cast

- É possível explicitamente fazer a conversão de tipos usando o operador *cast*.
- Sintaxe: `(tipo) expressão`

```
int a = 5, b = 2, c;
float d, e, f;
d = a/b;           //divisão com resultado inteiro
e = (float)a/b;    //divisão com resultado real
f = a/(float)b;    //divisão com resultado real
c = a/(int)f;      //divisão com resultado inteiro
printf("d=%f, e=%f, f=%f, c=%d", d, e, f, c);
```

```
d=2.000000, e=2.500000, f=2.500000, c=2
```

Impressão de dados

```
/* Programa que calcula
 * área de um círculo
 */
int main()
{
  //Declara variáveis
  float raio, area;

  //Imprime informações para o usuário
  printf(" Programa que calcula");
  printf(" area de um circulo.\n");
  printf(" Digite o raio: ");

  //Lê dado de entrada
  scanf("%f", &raio );

  //Calcula area
  area = 3.14159 * raio * raio;

  //Imprime resultado
  printf(" Area do circulo: %f", area);
  return 0;
}
```

`printf` pode conter:

- Expressão com comandos de formatação de tipos e códigos especiais
- Valores, variáveis ou expressões dos tipos indicados pelos comandos de formatação

Impressão de dados

Código	Tipo	Elemento armazenado
`%c`	`char`	um único caractere
`%d` ou `%i`	`int`	um inteiro
`%f`	`float`	um número em ponto flutuante
`%lf`	`double`	ponto flutuante com dupla precisão
`%e`	`float` ou `double`	um número na notação científica
`%s`	(tipo composto)	uma cadeia de caracteres

Impressão de valores reais

- Por padrão, a maioria dos compiladores C exibem os números de *ponto flutuante* com seis casas decimais.
- Para alterar este número podemos acrescentar **.n** ao código de formatação da saída, sendo *n* o número de casas decimais pretendido.

```c
#include <stdio.h>
int main()
{
  printf("Default: %f \n",3.1415169265);
  printf("Uma casa: %.1f \n",3.1415169265);
  printf("Duas casas: %.2f \n",3.1415169265);
  printf("Tres casas: %.3f \n",3.1415169265);
  printf("Notacao Cientifica: %e \n",3.1415169265);
  return 0;
}
```

Alinhamento de Saída

- O programa pode fixar a coluna da tela a partir da qual o conteúdo de uma variável, ou o valor de uma constante será exibido.
- Isto é obtido acrescentado-se um inteiro **m** ao código de formatação. Neste caso, *m* indicará o número de colunas que serão utilizadas para exibição do conteúdo.

```
#include <stdio.h>
int main()
{
  printf("Valor: %d \n",25);
  printf("Valor: %10d \n",25);
  return 0;
}
```

Impressão de Códigos Especiais

Código	Ação
`\n`	leva o cursor para a próxima linha
`\t`	executa uma tabulação
`\b`	executa um retrocesso
`\f`	leva o cursor para a próxima página
`\a`	emite um sinal sonoro (*beep*)
`\"`	exibe o caractere "
`\\`	exibe o caractere \
`%%`	exibe o caractere %

```c
#include <stdio.h>
int main()
{
  printf("\t\tx\n");
  printf("\tx\t\tx\n");
  printf("\t\tx\n");
  return 0;
}
```

Leitura de dados

```
/* Programa que calcula
 * área de um círculo
 */
int main()
{
  //Declara variáveis
  float raio, area;

  //Imprime informações para o usuário
  printf(" Programa que calcula");
  printf(" area de um circulo.\n");
  printf(" Digite o raio: ");

  //Lê dado de entrada
  scanf("%f", &raio );

  //Calcula area
  area = 3.14159 * raio * raio;

  //Imprime resultado
  printf(" Area do circulo: %f", area);
  return 0;
}
```

`scanf` deve conter:

- Expressão com comandos de identificação de tipos
- Endereço de variáveis para cada comando indicado na expressão

Comentários

```
/* Programa que calcula
 * área de um círculo
 */
int main()
{
  //Declara variáveis
  float raio, area;

  //Imprime informações para o usuário
  printf(" Programa que calcula");
  printf(" area de um circulo.\n");
  printf(" Digite o raio: ");

  //Lê dado de entrada
  scanf("%f", &raio );

  //Calcula area
  area = 3.14159 * raio * raio;

  //Imprime resultado
  printf(" Area do circulo: %f", area);
  return 0;
}
```

- Ignorados pelo compilador
- Auxiliam outros programadores a entender o código
- “ // ” faz com que o restante da linha seja ignorado
- “ /* ” faz com que todo conteúdo seja ignorado até que apareça “ */ ”

Importante

Para fazer os exercícios, é necessário:

1. Conhecer a sintaxe e semântica do conteúdo já abordado da linguagem C;
2. Entender o problema proposto;
3. Elaborar o algoritmo e escrever o programa;
4. Executar o programa com diferentes dados de entrada e verificar se este realmente resolve o problema proposto.

Exercícios

1. Escreva um programa que leia dois valores inteiros e efetue as seguintes operações matemáticas: adição, subtração, multiplicação, divisão e módulo (resto da divisão).
Para os valores de entrada 5 e 2, o programa deverá exibir na tela:

```
OPERACOES SOBRE INTEIROS
Digite o primeiro valor: 5
Digite o segundo valor: 2
5 + 2 = 7
5 - 2 = 3
5 * 2 = 10
5 / 2 = 2
5 % 2 = 1
```

Observe que:
Para imprimir o símbolo %, você precisa usar %%, como segue o exemplo: **`printf("%d %% %d = %d\n", ...`**

Exercícios

2. Elabore um programa que calcule o índice de massa corporal (IMC) de uma pessoa. Para isso, peça ao usuário para digitar seu peso (em Kg) e sua altura (em m), calcule o valor do seu IMC e imprima-o.

O IMC é determinado pela divisão do peso da pessoa pelo quadrado de sua altura.

Com peso de 61.5 Kg e altura de 1.70 m, o programa deve exibir:

```
INDICE DE MASSA CORPORAL
Digite o peso em kg: 61.5
Digite a altura em m: 1.70
Valor do IMC: 21.280276
```

Exercícios

3. Elabore um programa que imprima o extrato de uma conta salário que permite até 3 retiradas por mês. Leia o valor do saldo inicial, o valor do salário e o valor de cada retirada, imprimindo saldos parciais e final.
   Ao ser executado, o programa deve exibir:

```
EXTRATO BANCARIO
Saldo inicial: R$2000
Salario: R$400.3
Saldo parcial: R$2400.30
1a retirada: R$150
Saldo parcial: R$2250.30
2a retirada: R$250.3
Saldo parcial: R$2000.00
3a retirada: R$499.5
Saldo final: R$1500.50
```

DESAFIO: tente resolver o problema usando apenas duas variáveis.

Exercícios

4. Elabore um programa que resolva um sistema de equações composto por equações com a soma e a diferença entre dois números reais. Seu programa deve ler o total da soma e da diferença que aparecem nas equações, calcular os valores das variáveis e, ao final, imprimir a solução.
   Ao ser executado, o programa deve exibir:

```
SISTEMA DE EQUACOES
x + y = 14.8
x - y = 3.5
Solucao:
x = 9.150000
y = 5.650000
```

Exercícios

DESAFIO: Uma dona de casa precisa pagar a empregada doméstica e a babá e quer sair do banco apenas com a quantia necessária para pagá-las. O problema é que as funcionárias não podem dar troco, então ela precisa saber quantas notas de cada valor vai precisar para efetuar o pagamento.

Por exemplo, para pagar R$510,00 e R$490,00, não é suficiente ter 10 notas de R$100,00; são necessárias 9 notas de R$100,00, 1 nota de R$50,00, 2 notas de R$20,00 e 1 nota de R$10,00.

Faça um programa que leia o valor dos dois salários e calcule o número de notas necessárias para efetuar os pagamentos. A dona de casa não quer andar com moedas nem notas de R$2,00 e os salários devem ser arredondados para cima (um número múltiplo de 5).

```
PAGAMENTO SEM TROCO
1o valor: R$725
2o valor: R$443
Notas: 11xR$100;0xR$50;3xR$20;0xR$10;2xR$5.
```