Estruturas de Dados

Módulo 2 – Expressões

Avisos

- O ciclo básico alterou o horário da P2:
  - a P2 de ED será em 21/05, das 9h às 11h,
    e não das 11h às 13h, como no programa original

Referências

Waldemar Celes, Renato Cerqueira, José Lucas Rangel, *Introdução a Estruturas de Dados*, Editora Campus (2004)

Capítulo 2 – Expressões

Tópicos

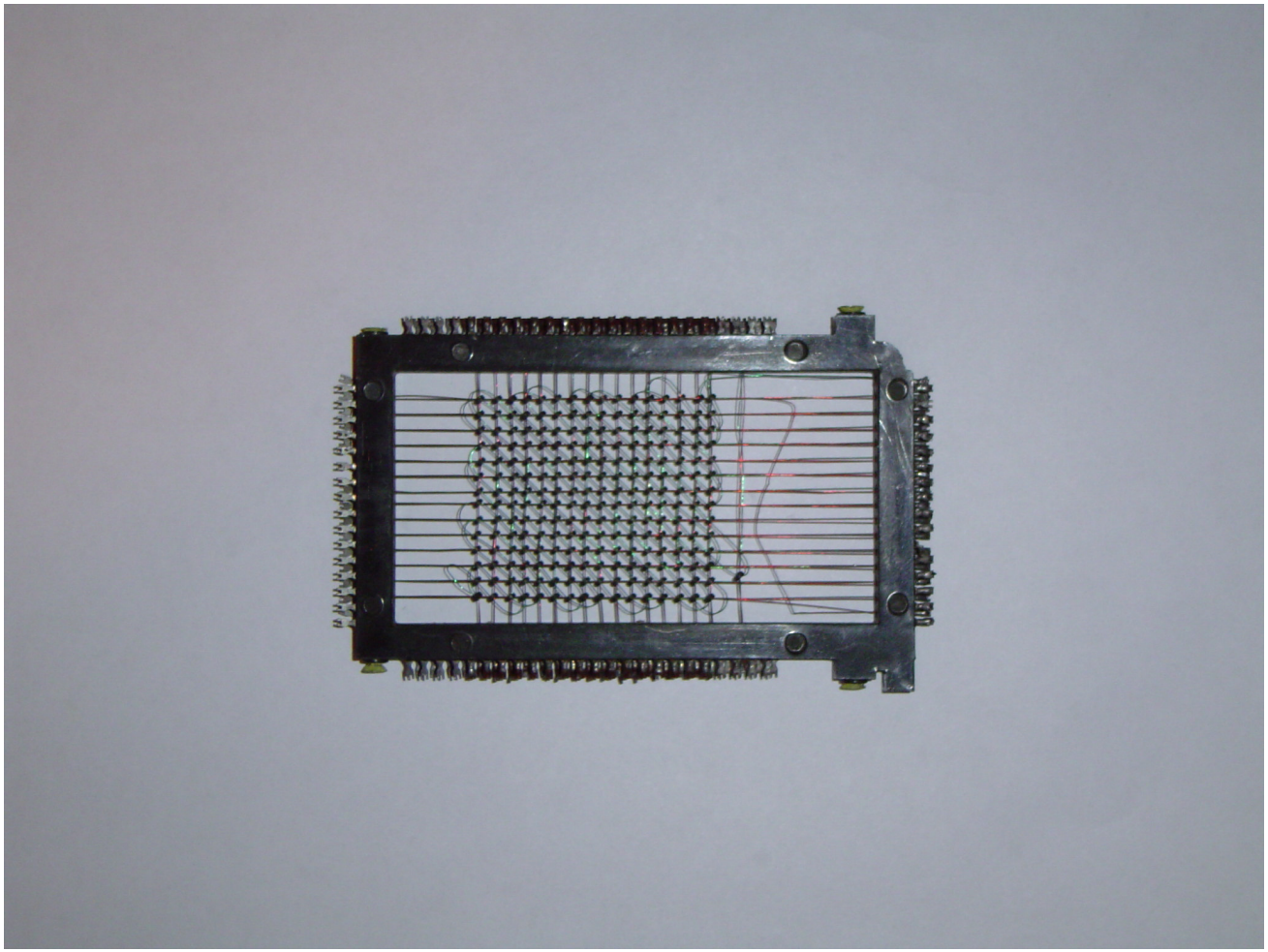
- Bits, Bytes e Palavras
- Variáveis e constantes
- Operadores e expressões

Bits, Bytes e Palavras

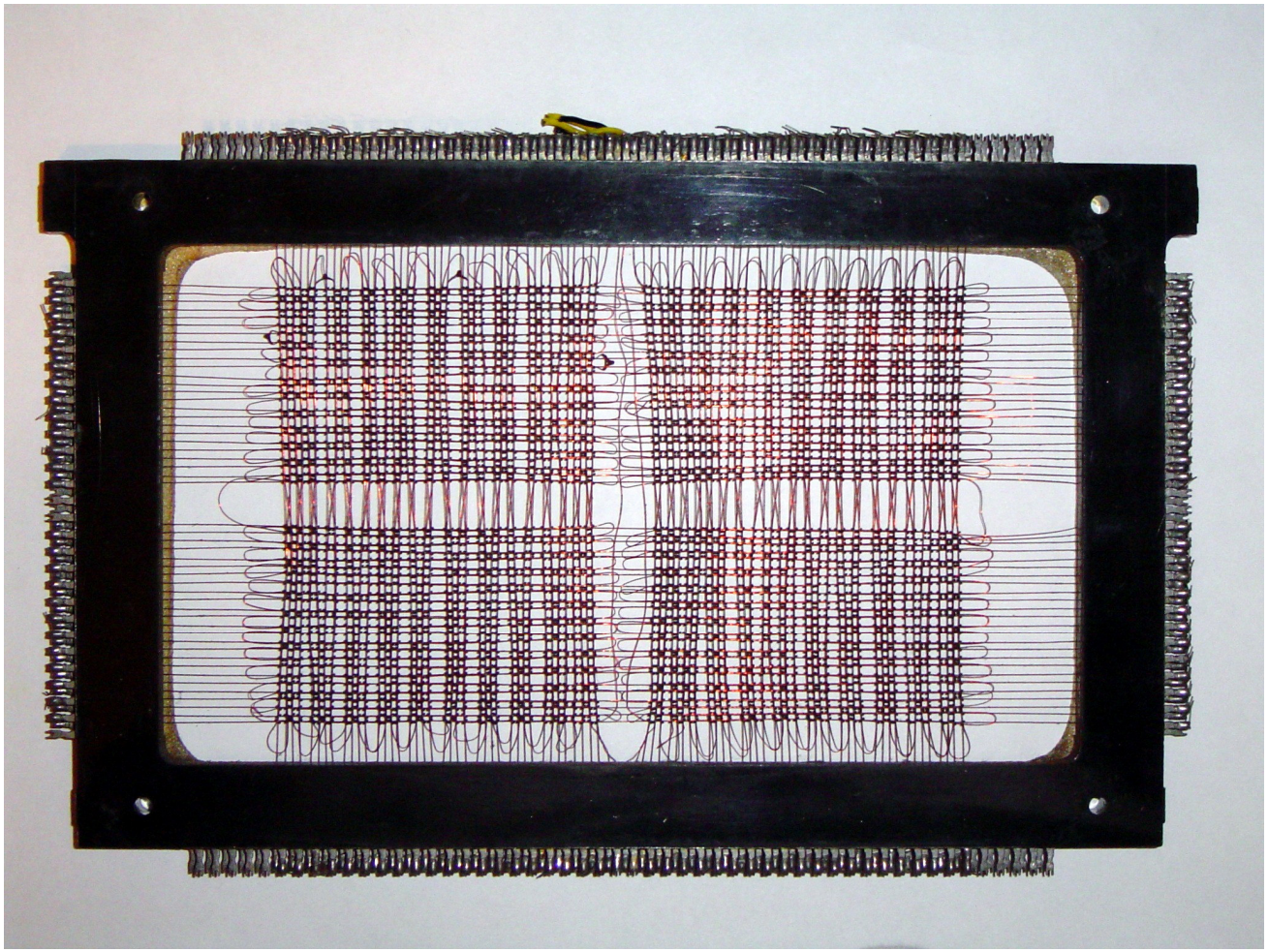
- Organização da memória
  - Bit:
    - menor unidade
    - armazena 0 ou 1
  - Byte:
    - seqüência de 8 bits
  - Palavra:
    - seqüência de bytes
    - número de bytes da palavra varia conforme a arquitetura do computador

		0	1	2	3	4	5	6	7
1	0	0	1	1	1	0	0	1	0
	1	1	1	0	0	1	1	1	0
	2	0	1	1	1	0	0	1	0
	3	0	0	0	0	0	0	0	0
2	0	1	1	1	0	1	0	1	0
	1	0	0	0	0	0	0	0	0
	2	1	1	1	1	1	1	1	1
	3	0	0	0	0	0	0	0	0

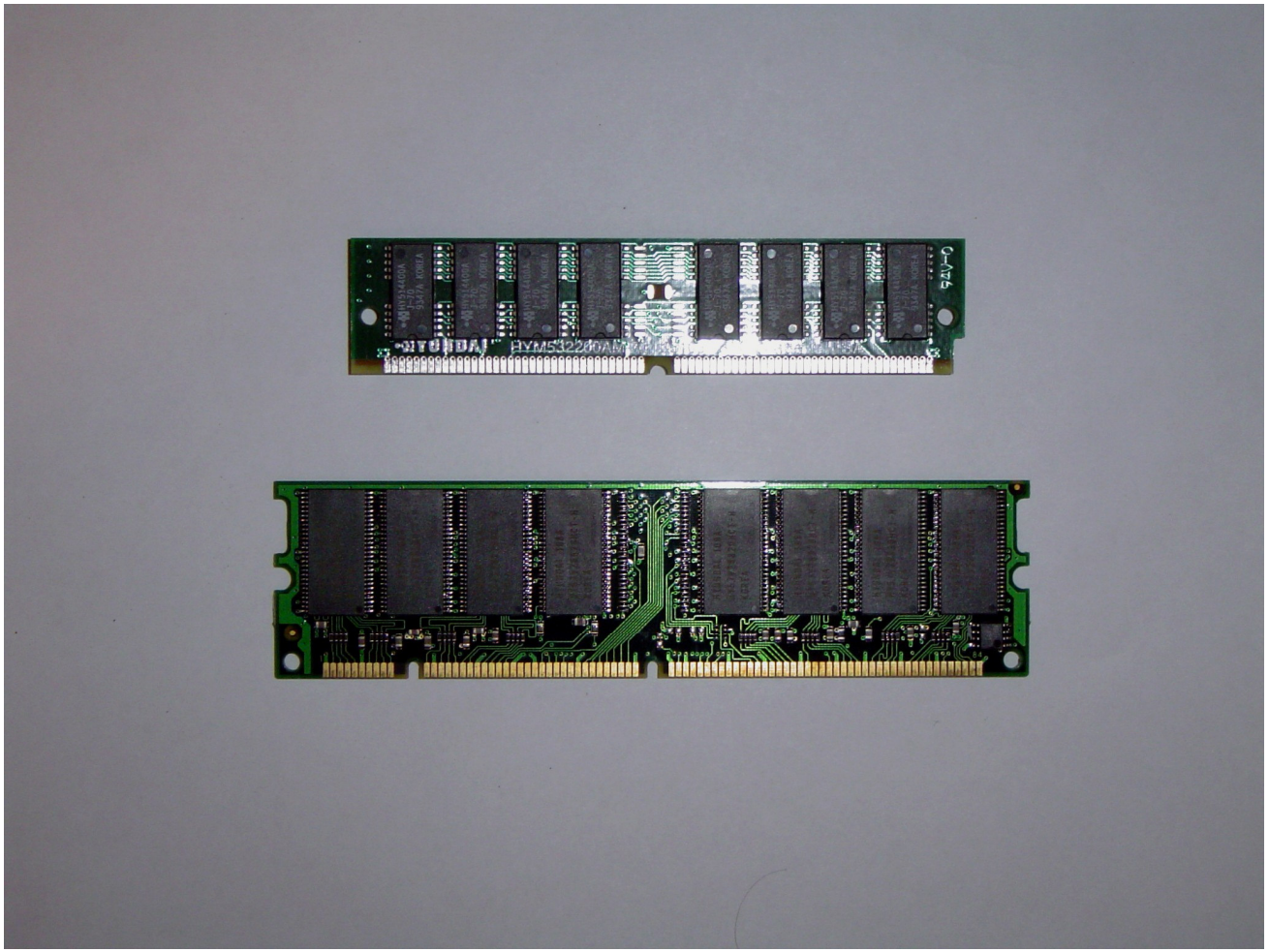
Bits, Bytes e Palavras



Bits, Bytes e Palavras



Bits, Bytes e Palavras



Variáveis e Constantes

Questão 1:

Suponha que:

a = 3

b = a / 2

c = b + 3.1

Qual é o valor de c?

- ❑ c = 4.6
- ❑ c = 4.1
- ❑ c = 4
- ❑ Nenhuma das opções acima
- ❑ Não é possível determinar o valor de c

Variáveis e Constantes

- Tipos básicos:

Tipo	Tamanho	Menor valor	Maior valor
char	1 byte	-128	+127
unsigned char	1 byte	0	+255
short int (short)	2 bytes	-32.768	+32.767
unsigned short int	2 bytes	0	+65.535
int (*)	4 bytes	-2.147.483.648	+2.147.483.647
long int (long)	4 bytes	-2.147.483.648	+2.147.483.647
unsigned long int	4 bytes	0	+4.294.967.295
float	4 bytes	-10^{38}	$+10^{38}$
double	8 bytes	-10^{308}	$+10^{308}$

(*) depende da máquina, sendo 4 bytes para arquiteturas de 32 bits

Variáveis e Constantes

- Valor Constante:
  - armazenado na memória
  - possui um tipo, indicado pela sintaxe da constante

```
123       /* constante inteira do tipo “int” */
12.45     /* constante real do tipo “double” */
1245e-2   /* constante real do tipo “double” */
12.45F    /* constante real do tipo “float” */
```

Variáveis e Constantes

- Variável:
  - espaço de memória para armazenar um dado
  - não é uma variável no sentido matemático
  - possui um tipo e um nome
    - nome: identifica o espaço de memória
    - tipo: determina a natureza do dado

Variáveis e Constantes

- Declaração de variável:
  - variáveis devem ser explicitamente declaradas
  - variáveis podem ser declaradas em conjunto

```
int a;      /* declara uma variável do tipo int */
int b;      /* declara uma variável do tipo int */
float c;    /* declara uma variável do tipo float */

int d, e;   /* declara duas variáveis do tipo int */
```

Variáveis e Constantes

- Declaração de variável:
  - variáveis só armazenam valores do mesmo tipo com que foram declaradas

```
int a;      /* declara uma variável do tipo int */
a = 4.3;    /* a armazenará o valor 4 */
```

Variáveis e Constantes

- Variável com valor indefinido:
  - uma variável pode receber um valor quando é definida (inicializada), ou através de um operador de atribuição

```
int a = 5, b = 10;      /* declara e inicializa duas variáveis do tipo int */
float c = 5.3;          /* declara e inicializa uma variável do tipo float */
```

Variáveis e Constantes

- Variável com valor indefinido:
  - uma variável deve ter um valor definido quando é utilizada

```
int a, b, c;          /* declara e inicializa duas variáveis do tipo int */
a = 2;
c = a + b;            /* ERRO: b contém “lixo” */
```

Operadores e Expressões

- Operadores:
  - aritméticos
  - atribuição
  - incremento e decremento
  - relacionais e lógicos
  - outros

Operadores e Expressões

- Operadores aritméticos (+ , - , * , / , %):
  - operações são feitas na precisão dos operandos
    - o operando com tipo de menor expressividade é convertido para o tipo do operando com tipo de maior expressividade
    - divisão entre inteiros trunca a parte fracionária

```
int a
double b, c;
a = 3.5;          /* a recebe o valor 3 */
b = a / 2.0;      /* b recebe o valor 1.5 */
c = 1/3 + b;      /* 1/3 retorna 0 pois a operação será sobre inteiros */
                  /* c recebe o valor de b */
```

Operadores e Expressões

- Operadores aritméticos (cont.):
  - o operador módulo, “%”, aplica-se a inteiros
  - precedência dos operadores: * , / , - , +

```
x % 2                  /* o resultado será 0, se x for par; caso contrário, será 1 */

a + b * c / d          é equivalente a      (a + ((b * c) / d))
```

Operadores e Expressões

- Operadores de atribuição (= , += , -= , *= , /= , %=):
  - C trata uma atribuição como uma expressão
    - a ordem é da direita para a esquerda
  - C oferece uma notação compacta para atribuições em que a mesma variável aparece dos dois lados

    var *op*= expr é equivalente a var = var *op* (expr)

i += 2;	é equivalente a	i = i + 2;
x *= y + 1;	é equivalente a	x = x * (y +1);

Operadores e Expressões

- Operadores de incremento e decremento (++ , --):
  - incrementa ou decrementa de uma unidade o valor de uma variável
    - os operadores não se aplicam a expressões
    - o incremento pode ser antes ou depois da variável ser utilizada

      n++ incrementa n de uma unidade, depois de ser usado

      ++n incrementa n de uma unidade, antes de ser usado

```
n = 5;
x = n++;            /* x recebe 5; n é incrementada para 6 */
x = ++n;            /* n é incrementada para 6; x recebe 6 */
a = 3;
b = a++ * 2;        / b termina com o valor 6 e a com o valor 4 */
```

Operadores e Expressões

- Operadores relacionais (< , <= , == , >= , > , !=):
  - o resultado será 0 ou 1 (não há valores booleanos em C)

```
int a, b;
int c = 23;
int d = c + 4;

c < 20              retorna 0
d > c               retorna 1
```

Operadores e Expressões

- Operadores lógicos (&& , || , !)
  - a avaliação é da esquerda para a direita
  - a avaliação pára quando o resultado pode ser conhecido

```
int a, b;
int c = 23;
int d = c + 4;

a = (c < 20) || (d > c);          /* retorna 1 */
                                  /* as duas sub-expressões são avaliadas */
b = (c < 20) && (d > c);          /* retorna 0 */
                                  /* apenas a primeira sub-expressão é avaliada */
```

Operadores e Expressões

- *sizeof:*
  - retorna o número de bytes ocupados por um tipo

```
int a = sizeof(float)                /* armazena 4 em a */
```

Operadores e Expressões

- conversão de tipo:
  - conversão de tipo é automática na avaliação de uma expressão
  - conversão de tipo pode ser requisita explicitamente

```
float f;              /* valor 3 é convertido automaticamente para “float”          */
float f = 3;          /* ou seja, passa a valer 3.0F, antes de ser atribuído a f    */

int g, h;             /* 3.5 é convertido (e arredondado) para “int”                */
g = (int) 3.5;        /* antes de ser atribuído à variável g                        */
h = (int) 3.5 % 2     /* e antes de aplicar o operador módulo “%”                   */
```

Exercício

- Defina as variáveis a, b e c para obter todas as possíveis respostas da Questão 1:

  Suponha que:

  a = 3

  b = a / 2

  c = b + 3.1

  Qual é o valor de c?

  - ❑ c = 4.6
  - ❑ c = 4.1
  - ❑ c = 4

Entrada e Saída

- Função “printf”:
  - possibilita a saída de valores segundo um determinado formato

```
printf (formato, lista de constantes/variáveis/expressões...);
```

```
printf ("%d %g", 33, 5.3);

tem como resultado a impressão da linha:
33 5.3
```

```
printf ("Inteiro = %d   Real = %g", 33, 5.3);

com saída:
Inteiro = 33   Real = 5.3
```

Entrada e Saída

- Especificação de formato:

`%c`	*especifica um char*
`%d`	*especifica um int*
`%u`	*especifica um unsigned int*
`%f`	*especifica um double (ou float)*
`%e`	*especifica um double (ou float) no formato científico*
`%g`	*especifica um double (ou float) no formato mais apropriado (%f ou %e)*
`%s`	*especifica uma cadeia de caracteres*

Entrada e Saída

- Impressão de texto:

```
printf("Curso de Estruturas de Dados\n");

exibe na tela a mensagem:
Curso de Estruturas de Dados
```

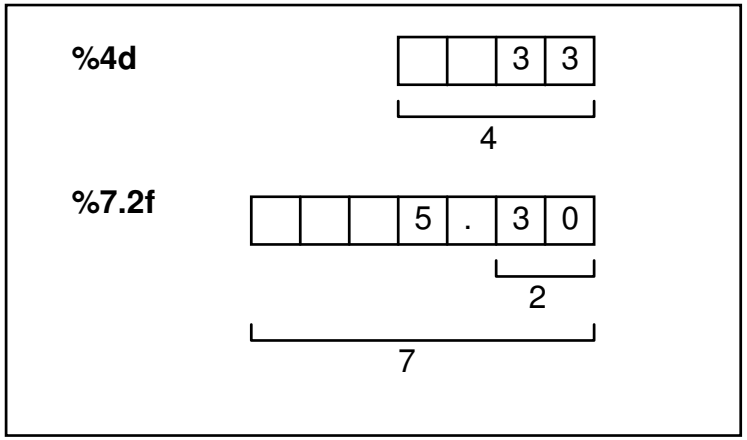
Entrada e Saída

- Especificação de caracteres de “escape”:

`\n`	*caractere de nova linha*
`\t`	*caractere de tabulação*
`\r`	*caractere de retrocesso*
`\"`	*caractere “*
`\\`	*caractere* \

Entrada e Saída

- Especificação de tamanho de campo:



Entrada e Saída

- Função “scanf”:
  - captura valores fornecidos via teclado

```
scanf (formato, lista de endereços das variáveis...);
```

```
int n;
scanf ("%d", &n);

valor inteiro digitado pelo usuário é armazenado na variável n
```

Entrada e Saída

- Especificação de formato:

`%c`	*especifica um char*
`%d`	*especifica um int*
`%u`	*especifica um unsigned int*
`%f,%e,%g`	*especificam um float*
`%lf, %le, %lg`	*especificam um double*
`%s`	*especifica uma cadeia de caracteres*

Entrada e Saída

- Função “scanf” (cont.):
  - caracteres diferentes dos especificadores no formato servem para cercar a entrada
  - espaço em branco dentro do formato faz com que sejam "pulados" eventuais brancos da entrada
  - %d, %f, %e e %g automaticamente pulam os brancos que precederem os valores numéricos a serem capturados

```
scanf ("%d:%d", &h, &m);

valores (inteiros) fornecidos devem ser separados pelo caractere dois pontos (:)
```