Algoritmos e Estruturas de Dados III

Aula 2.2 - Manipulação de dados em memória secundária

2023

PUC Minas

Roteiro da Aula

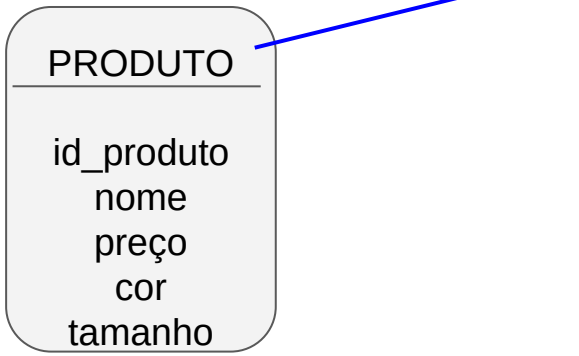
- Manipulando a Memória secundária
  - **Arquivos**
  - Fluxos de entrada e saída
  - Arquivos como vetores de bytes
  - CRUD (Implementando em JAVA)

Arquivos

Vamos pensar inicialmente em um contexto de um marketplace é um site de comércio eletrônico em que vendedores podem ofertar seus produtos para venda a consumidores finais

Arquivos

Vamos pensar inicialmente em um contexto de um marketplace é um site de comércio eletrônico em que vendedores podem ofertar seus produtos para venda a consumidores finais



ENTIDADE
Representação de um objeto físico ou abstrato em um sistema de dados

Arquivos

Vamos pensar inicialmente em um contexto de um marketplace é um site de comércio eletrônico em que vendedores podem ofertar seus produtos para venda a consumidores finais

ENTIDADE
Representação de um objeto físico ou abstrato em um sistema de dados

ATRIBUTO
Propriedade ou característica da entidade que ajuda a descrevê-la (no contexto do sistema de dados)

Arquivos

Vamos pensar inicialmente em um contexto de um marketplace é um site de comércio eletrônico em que vendedores podem ofertar seus produtos para venda a consumidores finais

ENTIDADE
Representação de um objeto físico ou abstrato em um sistema de dados

ATRIBUTO
Propriedade ou característica da entidade que ajuda a descrevê-la (no contexto do sistema de dados)

Arquivos

Terminologia

- **ENTIDADE**
  Coisa ou objeto do mundo real, que pode ser físico (ex.: pessoa, carro, produto, ...) ou abstrato (ex.: evento, cargo, matrícula, ...).

- **ATRIBUTO**
  Propriedade que ajuda a descrever uma entidade
  - Funcionário: nome, dataNascimento, endereço, telefone, email
  - Evento: descrição, data, horário, local, observações

Arquivos

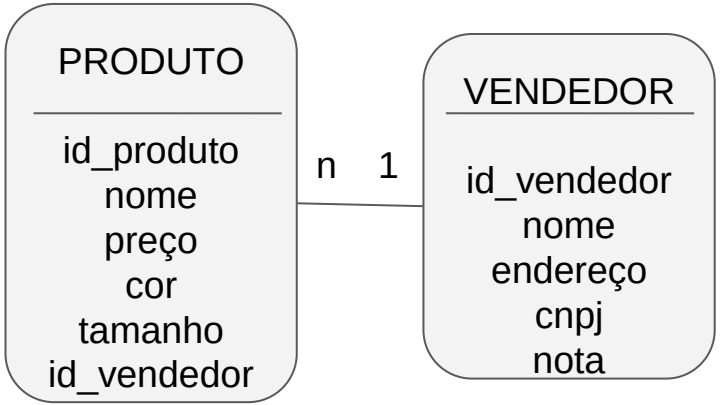
Vamos pensar inicialmente em um contexto de um marketplace é um site de comércio eletrônico em que vendedores podem ofertar seus produtos para venda a consumidores finais

Queremos acrescentar ao nosso sistema as lojas vendedoras.

Como ficará a representação?

Arquivos

Vamos pensar inicialmente em um contexto de um marketplace é um site de comércio eletrônico em que vendedores podem ofertar seus produtos para venda a consumidores finais

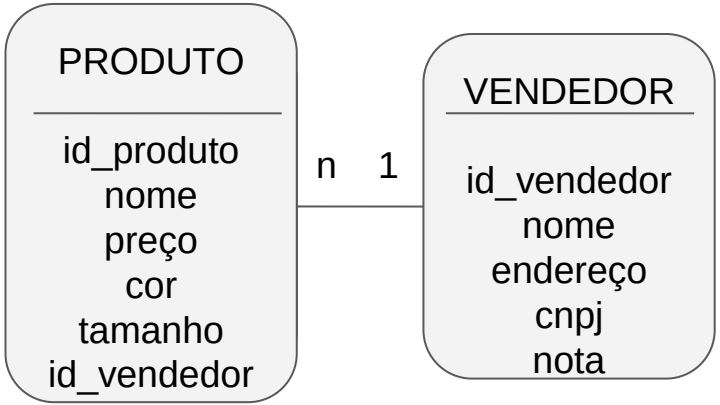


Queremos acrescentar ao nosso sistema as lojas vendedoras.

Como ficará a representação?

Arquivos

Vamos pensar inicialmente em um contexto de um marketplace é um site de comércio eletrônico em que vendedores podem ofertar seus produtos para venda a consumidores finais



Observações importantes:

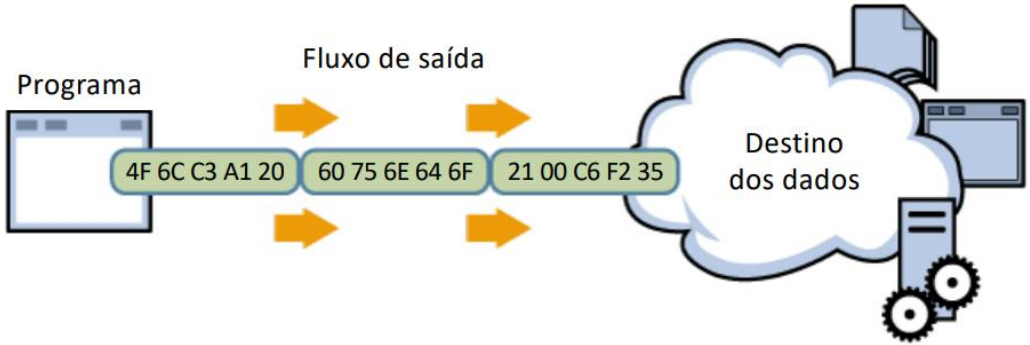
- Cada conjunto de entidades será armazenado em um arquivo diferente
- As operações básicas em arquivo são inclusão, leitura, alteração e exclusão (CRUD)

Roteiro da Aula

- Manipulando a Memória secundária
  - Arquivos
  - **Fluxos de entrada e saída**
  - Arquivos como vetores de bytes
  - CRUD (Implementando em JAVA)

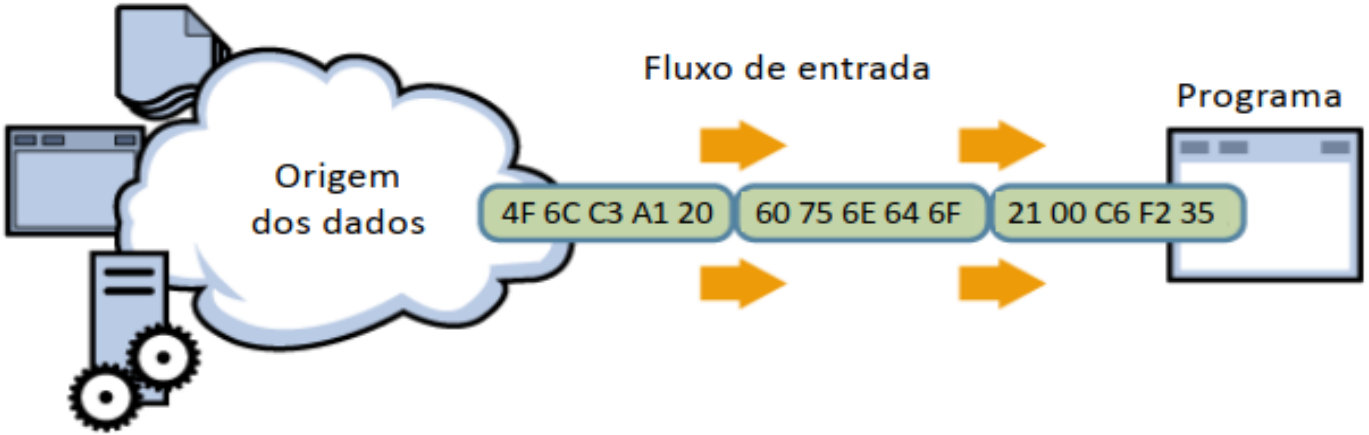
Fluxos de bytes

- A escrita de dados em um arquivo em disco é baseada no conceito de um fluxo:



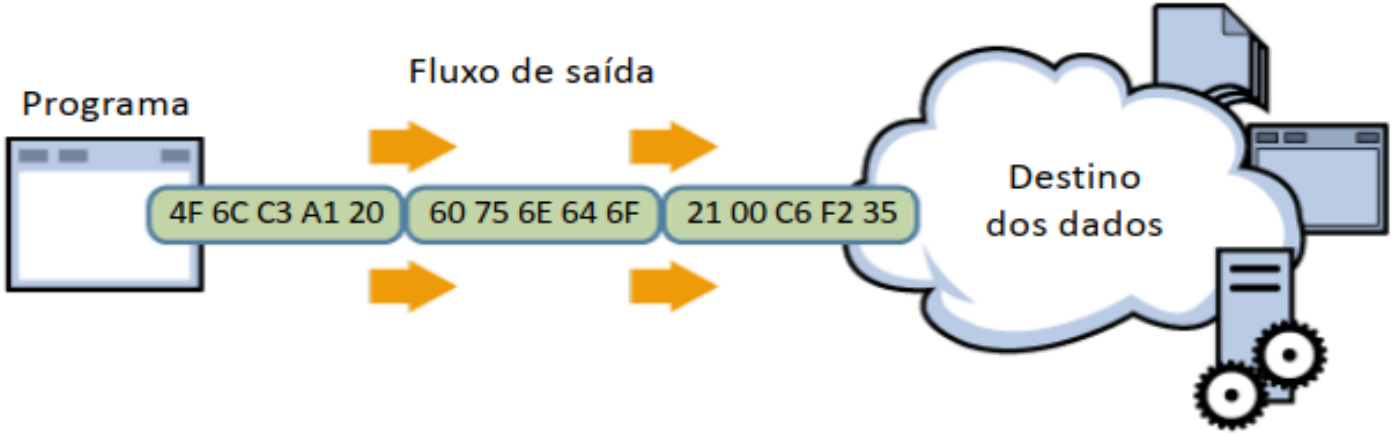
Os fluxos podem ser usados para escrita em disco, na rede, em memória, ...

Fluxos de entrada



Os fluxos podem ser usados para leitura do disco, da rede, da memória, ...

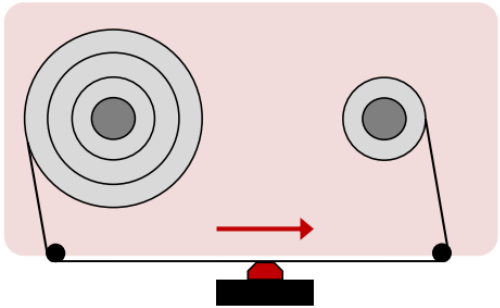
Fluxos de saída



Os fluxos podem ser usados para escrita em disco, na rede, em memória, ...

Fluxos de bytes

- A principal característica dos fluxos é que eles são **sequenciais**, isto é, não são otimizados para acesso aleatório.
- Apesar de obsoletas, as fitas são o melhor exemplo de dispositivo de acesso sequencial.



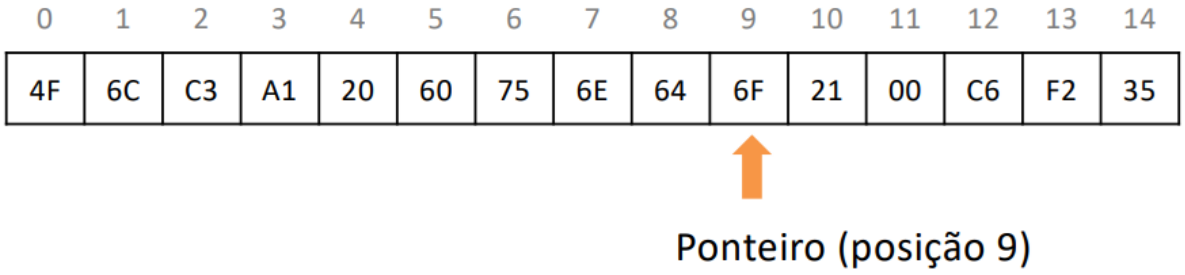
Roteiro da Aula

- Manipulando a Memória secundária
  - Arquivos
  - Fluxos de entrada e saída
  - **Arquivos como vetores de bytes**
  - CRUD (Implementando em JAVA)

Arquivos

O arquivo no disco pode ser visto como um vetor de bytes, com posições sequenciais iniciadas em zero.

- O ponto em que os dados serão lidos ou escritos é identificado por um ponteiro.



Arquivos

Quaisquer valores a serem escritos no arquivo devem ser convertidos em uma sequência de bytes.

int ID = 25

00	00	00	19

float pontos = 49.90

42	47	99	9A

String nome = "Conceição" (usando UTF-8)

com indicador de tamanho	00	0B	43	6F	6E	63	65	69	C3	A7	C3	A3	6F
com delimitador	43	6F	6E	63	65	69	C3	A7	C3	A3	6F	00	

$49_{10} = 110001_2$
$.90_{10} = 0.111001..._2$
$49.90_{10} = 110001.111001..._2$
$49.90_{10} = 1.10001111001..._2 \times 2^5$
Expoente = 132 (127+5)
Mantissa = 10001111001100110011001
49.90_{10} = 0100 0010 0100 0111 1001 1001 1001 1010 $_2$

Arquivos

Conversores online

Int
https://www.rapidtables.com/convert/number/decimal-to-hex.html
Float
https://onlineutf8tools.com/convert-utf8-to-hexadecimal
String
https://www.h-schmidt.net/FloatConverter/IEEE754.html

Exemplo de Arquivos de Jogadores

1º Jogador
```
int ID = 25;
String nome = "Conceição";
float pontos = 49.90;
```

2º Jogador
```
int ID = 37;
String nome = "José Carlos";
float pontos = 62.50;
```

3º Jogador
```
int ID = 291;
String nome = "Pedro";
float pontos = 53.45;
```

ENTIDADE
Representação de um objeto físico ou abstrato em um sistema de dados

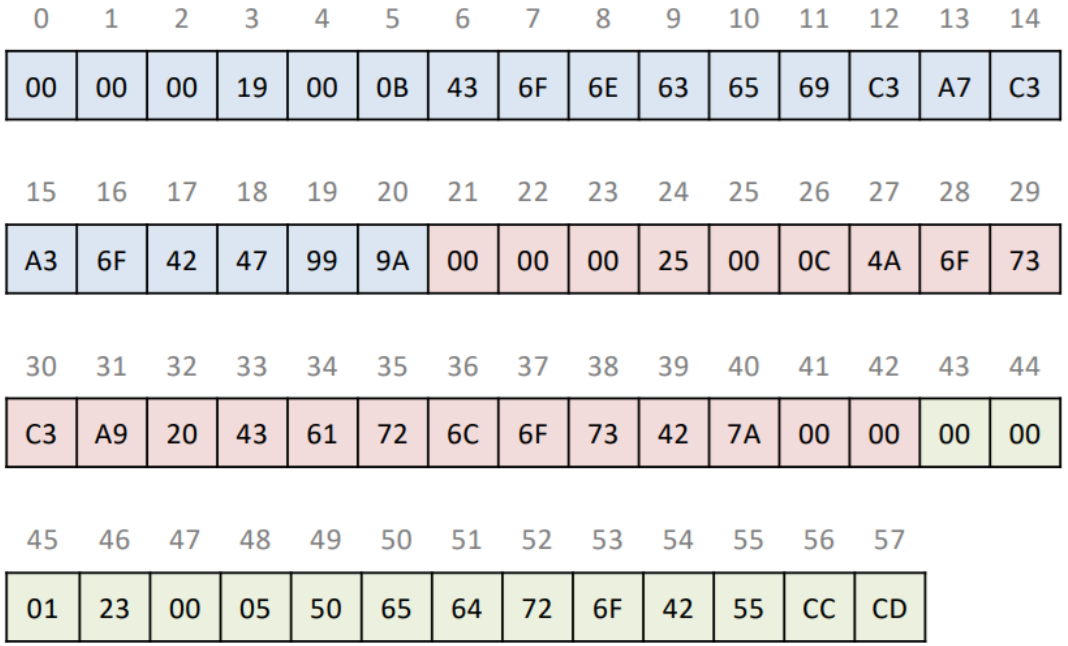
ATRIBUTOS
Propriedade ou característica da entidade que ajuda a descrevê-la (no contexto do sistema de dados)

Exemplo de Arquivos de Jogadores

1º Jogador
int ID = 25;
String nome = "Conceição";
float pontos = 49.90;

2º Jogador
int ID = 37;
String nome = "José Carlos";
float pontos = 62.50;

3º Jogador
int ID = 291;
String nome = "Pedro";
float pontos = 53.45;

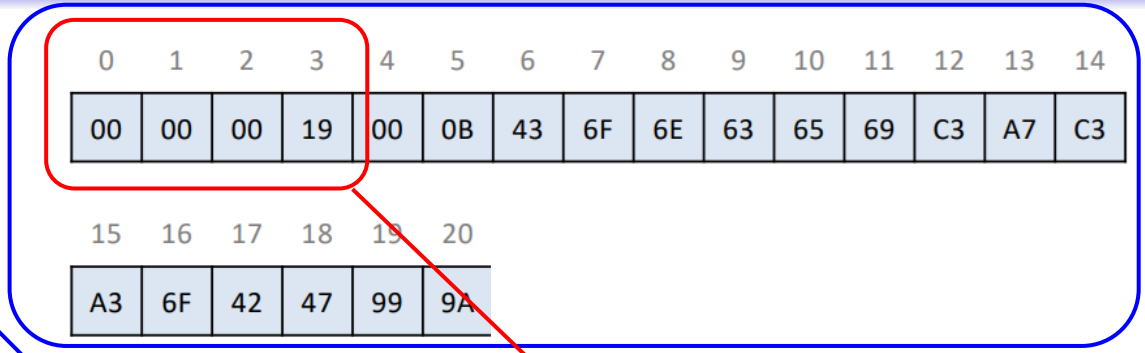


Exemplo de Arquivos de Jogadores

1º Jogador
int ID = 25;
String nome = "Conceição";
float pontos = 49.90;

2º Jogador
int ID = 37;
String nome = "José Carlos";
float pontos = 62.50;

3º Jogador
int ID = 291;
String nome = "Pedro";
float pontos = 53.45;



ENTIDADE

REGISTRO

CAMPO

ATRIBUTOS

Terminologia

ENTIDADE

Coisa ou **objeto do mundo real**, que pode ser físico (ex.: pessoa, carro, produto, ...) ou abstrato (ex.: evento, cargo, matrícula, ...).

ATRIBUTO

Propriedade que ajuda a **descrever uma entidade**

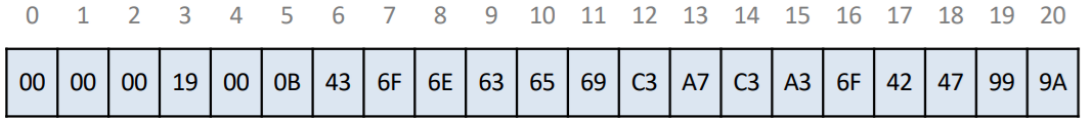
- Funcionário: nome, dataNascimento, endereço, telefone, email
- Evento: descrição, data, horário, local, observações

Terminologia

REGISTRO

Sequência de bytes em um arquivo que contém uma entidade

Jogador
int ID = 25;
String nome = "Conceição";
float pontos = 49.90;



O registro pode ter diferenças em relação à entidade

- Os dados do registro podem estar compactados ou criptografados
- O registro pode conter metadados (como a data da última atualização ou o campo lápide)
- O registro pode conter informações de encadeamento interno ou externo
- O registro pode conter informações de relacionamento entre entidades

Terminologia

CAMPO

Sequência de bytes em um arquivo que contém um atributo

int ID = 25

00	00	00	19

float pontos = 49.90

42	47	99	9A

String nome = "Conceição" (usando UTF-8)

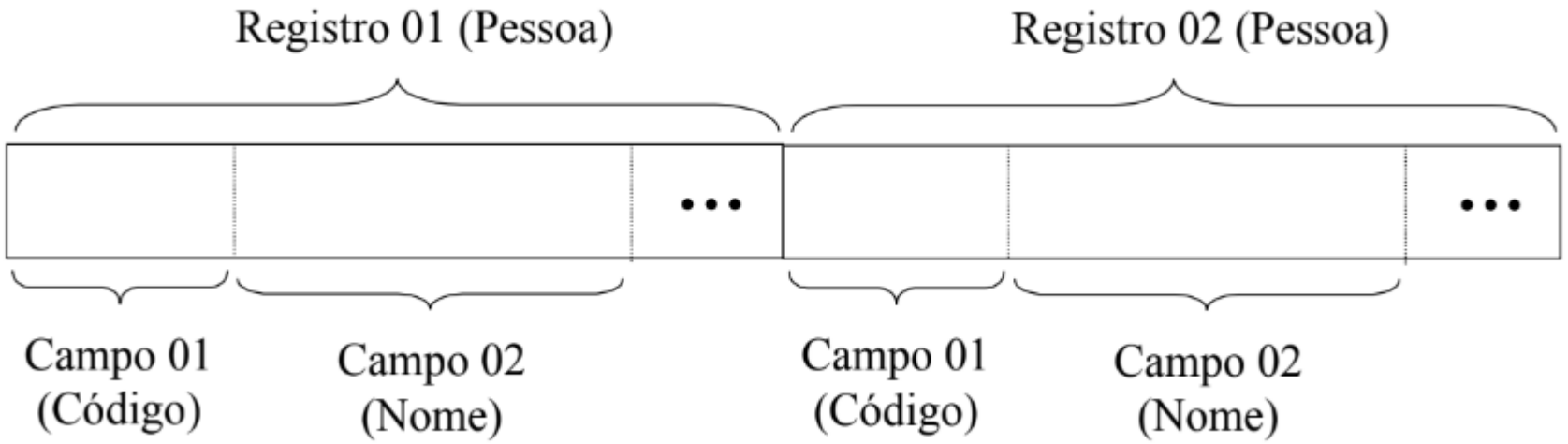
com indicador de tamanho

00	0B	43	6F	6E	63	65	69	C3	A7	C3	A3	6F

com delimitador

43	6F	6E	63	65	69	C3	A7	C3	A3	6F	00

Exemplo de Registro e Campo



00000000 00000000 00000000 00011001

1º Jogador

int ID = 25;

String nome = "Conceição";

float pontos = 49.90;

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
00	00	00	19	00	0B	43	6F	6E	63	65	69	C3	A7	C3

15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
A3	6F	42	47	99	9A	00	00	00	25	00	0C	4A	6F	73

2º Jogador

int ID = 37;

String nome = "José Carlos";

float pontos = 62.50;

30	31	32	33	34	35	36	37	38	39	40	41	42	43	44
C3	A9	20	43	61	72	6C	6F	73	42	7A	00	00	00	00

3º Jogador

int ID = 291;

String nome = "Pedro";

float pontos = 53.45;

45	46	47	48	49	50	51	52	53	54	55	56	57
01	23	00	05	50	65	64	72	6F	42	55	CC	CD

Exemplo de Arquivos de Jogadores

1º Jogador
int ID = 25;
String nome = "Conceição";
float pontos = 49.90;

2º Jogador
int ID = 37;
String nome = "José Carlos";
float pontos = 62.50;

3º Jogador
int ID = 291;
String nome = "Pedro";
float pontos = 53.45;

Conteúdo real do arquivo:

```
00 00 00 19 20 43 6F
6E 63 65 69 C3 A7 C3
A3 6F 41 47 99 9A 00
00 00 25 4A 6F 73 C3
A9 20 43 61 72 6C 6F
73 42 7A 00 00 00 00
01 23 50 64 64 72 6F
42 55 CC CD
```

Exemplo de Arquivos de Jogadores

1º Jogador
```
int ID = 25;
String nome = "Conceição";
float pontos = 49.90;
```

2º Jogador
```
int ID = 37;
String nome = "José Carlos";
float pontos = 62.50;
```

3º Jogador
```
int ID = 291;
String nome = "Pedro";
float pontos = 53.45;
```

Conteúdo real do arquivo:

```
00 00 00 19 20 43 6F
6E 63 65 69 C3 A7 C3
```

Se não houver (no código ou em outro lugar) uma descrição do que os bytes significam, esse arquivo tanto poderia ser uma lista de jogadores, quanto uma foto, um trecho de vídeo, uma planilha ou qualquer outra coisa.

Convertendo tipos primitivos para bytes

Tipo	Descrição	Valores
byte	Número inteiro de 8 bits com sinal	-128 a 127
short	Número inteiro de 16 bits com sinal	-32.768 a +32.767
int	Número inteiro de 32 bits com sinal	-2.147.483.648 a 2.147.483.647
long	Número inteiro de 64 bits com sinal	-9.223.372.036.854.775.808 a +9.223.372.036.854.775.807
float	Número de ponto flutuante de 32 bits com sinal	±1,40129846e-45 a ±3,40282347e+38
double	Número de ponto flutuante de 64 bits com sinal	±4,94065645841246544e-324 a ±1,79769313486231570e+308
char	Caractere ASCII (1 byte) ou Unicode de 16 bits (2 bytes)	*Qualquer caractere*
boolean	Variável lógica que indica falso ou verdadeiro (1 byte)	true ou false

https://conversoes.kutova.repl.co/

Convertendo tipos primitivos para bytes

Tipo	Descrição	Valores
byte	Número inteiro de 8 bits com sinal	-128 a 127
short	Número inteiro de 16 bits com sinal	-32.768 a +32.767
int	Número inteiro de 32 bits com sinal	-2.147.483.648 a 2.147.483.647
long	Número inteiro de 64 bits com sinal	-9.223.372.036.854.775.808 a +9.223.372.036.854.775.807
float	Número de ponto flutuante de 32 bits com sinal	±1,40129846e-45 a ±3,40282347e+38
double	Número de ponto flutuante de 64 bits com sinal	±4,94065645841246544e-324 a ±1,79769313486231570e+308
char	Caractere ASCII (1 byte) ou Unicode de 16 bits (2 bytes)	*Qualquer caractere*
boolean	Variável lógica que indica falso ou verdadeiro (1 byte)	true ou false

ALÉM DOS TIPOS PRIMITIVOS: string é representada como uma sequência de caracteres de tamanho variável em UTF-8, usando indicador de tamanho ou caractere delimitador

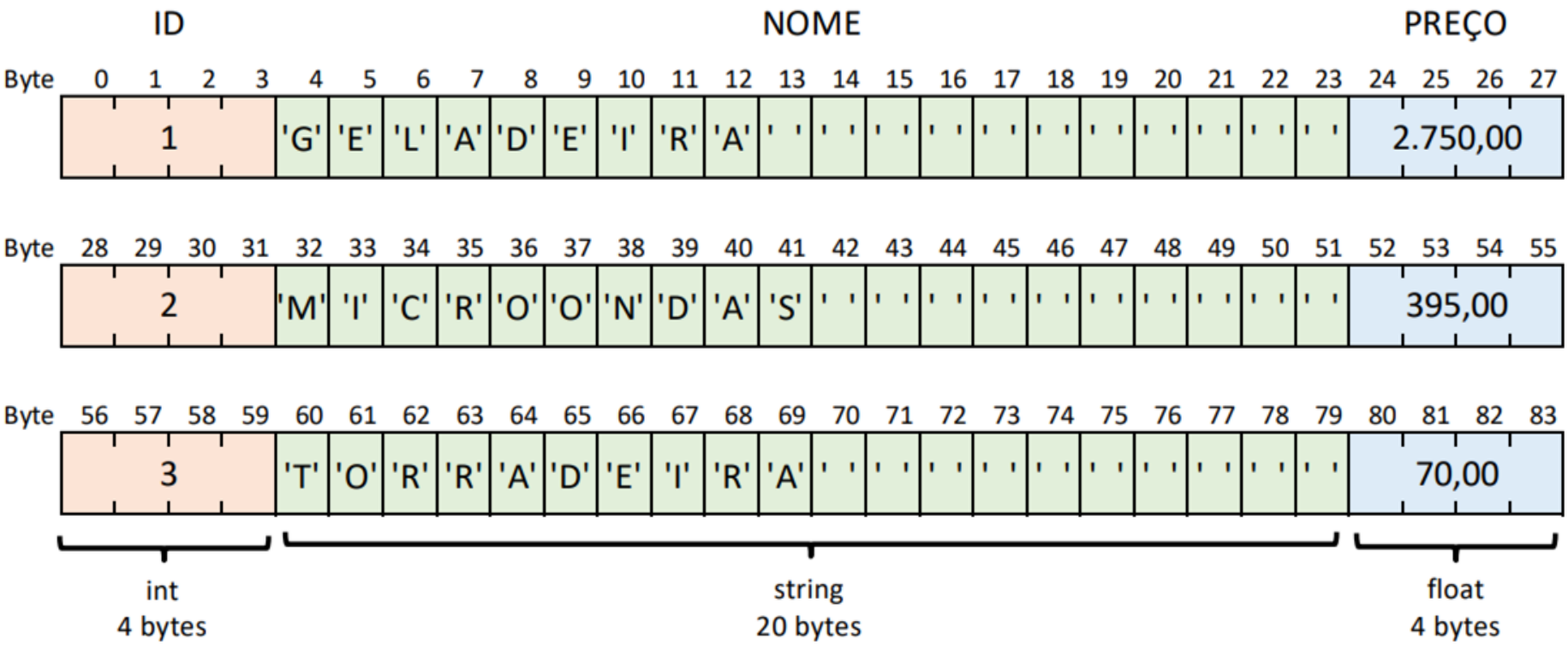
Convertendo Strings em bytes

C	o	n	c	e	i	ç	ã	o
43	6F	6E	63	65	69	C3A7	C3A3	6F

9 caracteres – 11 bytes

3 opções:

- **Tamanho fixo da String**
- Usar separadores
- Tamanho variável da String (precisa de 2 bytes para tamanho da String)


ID
NOME
PREÇO
Byte 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
1
'G' 'E' 'L' 'A' 'D' 'E' 'I' 'R' 'A' '
2.750,00
Byte 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
2
'M' 'I' 'C' 'R' 'O' 'O' 'N' 'D' 'A' 'S' '
395,00
Byte 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83
3
'T' 'O' 'R' 'R' 'A' 'D' 'E' 'I' 'R' 'A' '
70,00
int
4 bytes
string
20 bytes
float
4 bytes

Convertendo Strings em bytes

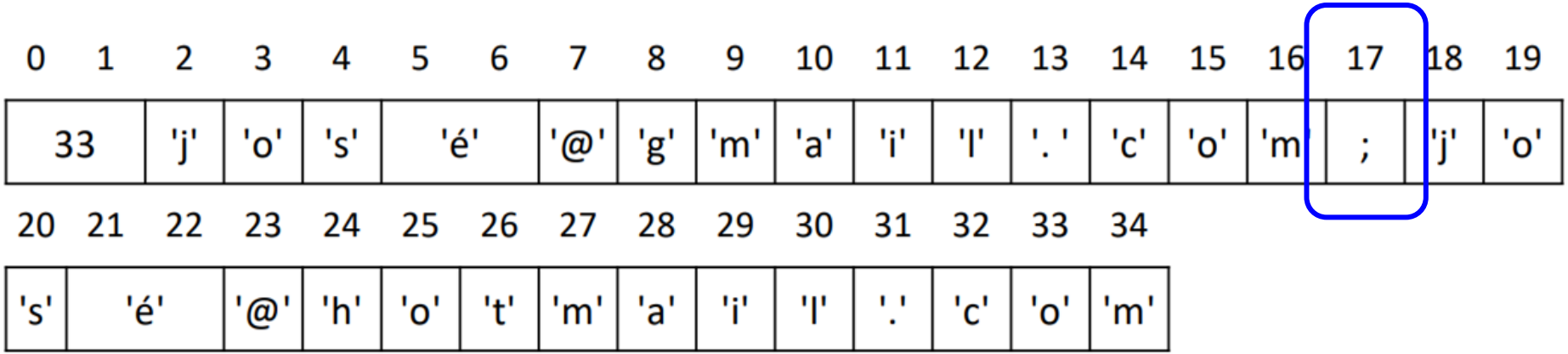
C	o	n	c	e	i	ç	ã	o	
43	6F	6E	63	65	69	C3A7	C3A3	6F	;

9 caracteres – 11 bytes

2 opções:

- Tamanho fixo da String
- **Usar separadores**
- Tamanho variável da String (precisa de 2 bytes para tamanho da String)

Convertendo Strings em bytes



Convertendo Strings em bytes

Taman.		C	o	n	c	e	i	ç	ã	o
00	0B	43	6F	6E	63	65	69	C3A7	C3A3	6F

9 caracteres – 11 bytes

2 opções:

- Tamanho fixo da String
- Usar separadores
- **Tamanho variável da String (precisa de 2 bytes para tamanho da String)**

- Com indicador de tamanho

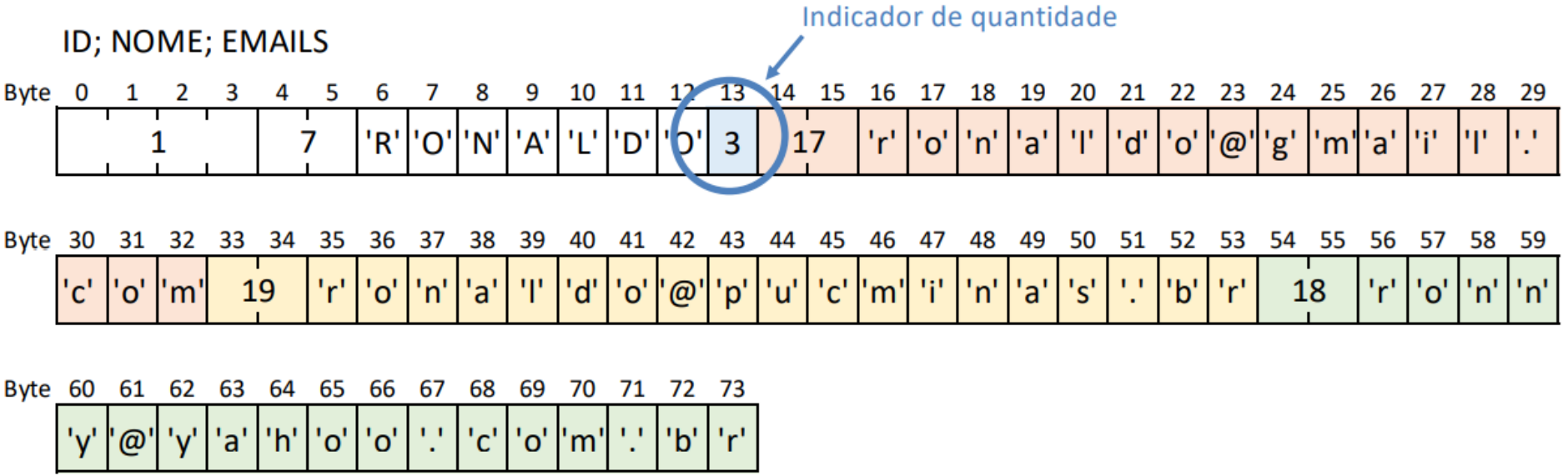
IND.TAM.		C	o	n	c	e	i	ç		ã		o
00	0B	43	6F	6E	63	65	69	C3	A7	C3	A3	6F

- Com delimitador

C	o	n	c	e	i	ç		ã		o	\n
43	6F	6E	63	65	69	C3	A7	C3	A3	6F	0A

  - Delimitadores tradicionais: \n \0 ; | (e outros)

- Exemplo – emails de um cliente:



Observações Importantes

• Valores numéricos não devem ser armazenados como strings (ex.: “11”), mas como tipos primitivos: byte, short, int, long, float e double.

• Nos tipos numéricos, não use mais bytes do que o necessário

(ex.: byte para número de dependentes, float para preços de livros)

• Cuidado com os falsos números (CPF, CNPJ, CEP, Telefone, ...)

Codificação de Caracteres

- Representação binária (ou hexa) de cada símbolo
- ASCII (anos 1960) - American Standard Code for Information Interchange
- Representação de 128 símbolos (7 bits)
- 100 0001 = 65 = 0x41 = 'A'
- O 8o bit (mais significativo) era usado como bit de paridade nas comunicações
- Faltavam os caracteres latinos acentuados, os chineses, os cirílicos (russos), ...

Codificação de Caracteres

ASCII

0	*null*	16	*data link escape*	32	*space*	48	0	64	@	80	P	96	`	112	p
1	*start of heading*	17	*device control 1*	33	!	49	1	65	A	81	Q	97	a	113	q
2	*start of text*	18	*device control 2*	34	"	50	2	66	B	82	R	98	b	114	r
3	*end of text*	19	*device control 3*	35	#	51	3	67	C	83	S	99	c	115	s
4	*end of transmission*	20	*device control 4*	36	$	52	4	68	D	84	T	100	d	116	t
5	*enquiry*	21	*negative acknowledge*	37	%	53	5	69	E	85	U	101	e	117	u
6	*ackknowledge*	22	*synchoronous idle*	38	&	54	6	70	F	86	V	102	f	118	v
7	*bell*	23	*end of trans. block*	39	'	55	7	71	G	87	W	103	g	119	w
8	*backspace*	24	*cancel*	40	(	56	8	72	H	88	X	104	h	120	x
9	*horizontal tab*	25	*end of medium*	41	)	57	9	73	I	89	Y	105	i	121	y
10	*line feed*	26	*substitute*	42	*	58	:	74	J	90	Z	106	j	122	z
11	*vertical tab*	27	*escape*	43	+	59	;	75	K	91	[	107	k	123	{
12	*form feed*	28	*file separator*	44	,	60	<	76	L	92	\	108	l	124	|
13	*carriage return*	29	*group separator*	45	-	61	=	77	M	93	]	109	m	125	}
14	*shift out*	30	*record separator*	46	.	62	>	78	N	94	^	110	n	126	~
15	*shift in*	31	*unit separator*	47	/	63	?	79	O	95	_	111	o	127	*del*

Codificação de Caracteres

Codificações de 8 bits

ISO-8859-1 (caracteres latinos)

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
8_	U+0080 PAD 128	U+0081 HOP 129	U+0082 BPH 130	U+0083 NBH 131	U+0084 IND 132	U+0085 NEL 133	U+0086 SSA 134	U+0087 ESA 135	U+0088 HTS 136	U+0089 HTJ 137	U+008A LTS 138	U+008B PLD 139	U+008C PLU 140	U+008D RI 141	U+008E SS2 142	U+008F SS3 143
9_	U+0090 DCS 144	U+0091 PU1 145	U+0092 PU2 146	U+0093 STS 147	U+0094 CCH 148	U+0095 MW 149	U+0096 SPA 150	U+0097 EPA 151	U+0098 SOS 152	U+0099 SGCI 153	U+009A SCI 154	U+009B CSI 155	U+009C ST 156	U+009D OSC 157	U+009E PM 158	U+009F APC 159
A_	U+00A0 NBSP 160	U+00A1 ¡ 161	U+00A2 ¢ 162	U+00A3 £ 163	U+00A4 ¤ 164	U+00A5 ¥ 165	U+00A6 ¦ 166	U+00A7 § 167	U+00A8 ¨ 168	U+00A9 © 169	U+00AA ª 170	U+00AB « 171	U+00AC ¬ 172	U+00AD 173	U+00AE ® 174	U+00AF ¯ 175
B_	U+00B0 ° 176	U+00B1 ± 177	U+00B2 ² 178	U+00B3 ³ 179	U+00B4 ´ 180	U+00B5 µ 181	U+00B6 ¶ 182	U+00B7 · 183	U+00B8 ¸ 184	U+00B9 ¹ 185	U+00BA º 186	U+00BB » 187	U+00BC ¼ 188	U+00BD ½ 189	U+00BE ¾ 190	U+00BF ¿ 191
C_	U+00C0 À 192	U+00C1 Á 193	U+00C2 Â 194	U+00C3 Ã 195	U+00C4 Ä 196	U+00C5 Å 197	U+00C6 Æ 198	U+00C7 Ç 199	U+00C8 È 200	U+00C9 É 201	U+00CA Ê 202	U+00CB Ë 203	U+00CC Ì 204	U+00CD Í 205	U+00CE Î 206	U+00CF Ï 207
D_	U+00D0 Ð 208	U+00D1 Ñ 209	U+00D2 Ò 210	U+00D3 Ó 211	U+00D4 Ô 212	U+00D5 Õ 213	U+00D6 Ö 214	U+00D7 × 215	U+00D8 Ø 216	U+00D9 Ù 217	U+00DA Ú 218	U+00DB Û 219	U+00DC Ü 220	U+00DD Ý 221	U+00DE Þ 222	U+00DF ß 223
E_	U+00E0 à 224	U+00E1 á 225	U+00E2 â 226	U+00E3 ã 227	U+00E4 ä 228	U+00E5 å 229	U+00E6 æ 230	U+00E7 ç 231	U+00E8 è 232	U+00E9 é 233	U+00EA ê 234	U+00EB ë 235	U+00EC ì 236	U+00ED í 237	U+00EE î 238	U+00EF ï 239
F_	U+00F0 ð 240	U+00F1 ñ 241	U+00F2 ò 242	U+00F3 ó 243	U+00F4 ô 244	U+00F5 õ 245	U+00F6 ö 246	U+00F7 ÷ 247	U+00F8 ø 248	U+00F9 ù 249	U+00FA ú 250	U+00FB û 251	U+00FC ü 252	U+00FD ý 253	U+00FE þ 254	U+00FF ÿ 255

Codificação de Caracteres

Codificações de 8 bits

ISO-8859-7 (caracteres gregos)

	_0	_1	_2	_3	_4	_5	_6	_7	_8	_9	_A	_B	_C	_D	_E	_F
8_	U+0080 PAD 128	U+0081 HOP 129	U+0082 BPH 130	U+0083 NBH 131	U+0084 IND 132	U+0085 NEL 133	U+0086 SSA 134	U+0087 ESA 135	U+0088 HTS 136	U+0089 HTJ 137	U+008A LTS 138	U+008B PLD 139	U+008C PLU 140	U+008D RI 141	U+008E SS2 142	U+008F SS3 143
9_	U+0090 DCS 144	U+0091 PU1 145	U+0092 PU2 146	U+0093 STS 147	U+0094 CCH 148	U+0095 MW 149	U+0096 SPA 150	U+0097 EPA 151	U+0098 SOS 152	U+0099 SGCI 153	U+009A SCI 154	U+009B CSI 155	U+009C ST 156	U+009D OSC 157	U+009E PM 158	U+009F APC 159
A_	U+00A0 NBSP 160	U+2018 ‘ 161	U+2019 ’ 162	U+00A3 £ 163	U+20AC € 164	U+20AF ₯ 165	U+00A6 ¦ 166	U+00A7 § 167	U+00A8 ¨ 168	U+00A9 © 169	U+037A ͺ 170	U+00AB « 171	U+00AC ¬ 172	U+00AD 173		U+2015 ― 175
B_	U+00B0 ° 176	U+00B1 ± 177	U+00B2 ² 178	U+00B3 ³ 179	U+0384 ΄ 180	U+0385 ΅ 181	U+0386 Ά 182	U+00B7 · 183	U+0388 Έ 184	U+0389 Ή 185	U+038A Ί 186	U+00BB » 187	U+038C Ό 188	U+00BD ½ 189	U+038E Ύ 190	U+038F Ώ 191
C_	U+0390 ΐ 192	U+0391 Α 193	U+0392 Β 194	U+0393 Γ 195	U+0394 Δ 196	U+0395 Ε 197	U+0396 Ζ 198	U+0397 Η 199	U+0398 Θ 200	U+0399 Ι 201	U+039A Κ 202	U+039B Λ 203	U+039C Μ 204	U+039D Ν 205	U+039E Ξ 206	U+039F Ο 207
D_	U+03A0 Π 208	U+03A1 Ρ 209		U+03A3 Σ 211	U+03A4 Τ 212	U+03A5 Υ 213	U+03A6 Φ 214	U+03A7 Χ 215	U+03A8 Ψ 216	U+03A9 Ω 217	U+03AA Ϊ 218	U+03AB Ϋ 219	U+03AC ά 220	U+03AD έ 221	U+03AE ή 222	U+03AF ί 223
E_	U+03B0 ΰ 224	U+03B1 α 225	U+03B2 β 226	U+03B3 γ 227	U+03B4 δ 228	U+03B5 ε 229	U+03B6 ζ 230	U+03B7 η 231	U+03B8 θ 232	U+03B9 ι 233	U+03BA κ 234	U+03BB λ 235	U+03BC μ 236	U+03BD ν 237	U+03BE ξ 238	U+03BF ο 239
F_	U+03C0 π 240	U+03C1 ρ 241	U+03C2 ς 242	U+03C3 σ 243	U+03C4 τ 244	U+03C5 υ 245	U+03C6 φ 246	U+03C7 χ 247	U+03C8 ψ 248	U+03C9 ω 249	U+03CA ϊ 250	U+03CB ϋ 251	U+03CC ό 252	U+03CD ύ 253	U+03CE ώ 254	255

Codificação de Caracteres

Unicode

• Sistema de codificação de caracteres capaz de representar texto de qualquer sistema de escrita existente (mais de 100 mil símbolos)
• Sistemas latino, arábico, cirílico, chinês, hebráico, ...
• Símbolos matemáticos, geométricos, musicais, setas, ícones, emojis, ...
• Escrita cuneiforme, Braille, runas, élfico, ...
• Próximos: hieróglifos egípcios, alfabeto babilônico, ...

Referências:
• http://unicode.org/
• http://www.unicodetables.com/

Codificação de Caracteres

Unicode

UTF-8 (8-bit Unicode Transformation Format)
• Codificação Unicode de comprimento variável (1 a 4 bytes), que pode representar qualquer caráter (code point) do padrão.
• Padrão usado na Internet e na Web

UTF-16
• Codificação de 2 ou 4 bytes, usada principalmente para escrita em idiomas dos países asiáticos

UTF-32
• Codificação de comprimento fixo de 4 bytes

Codificação de Caracteres

UTF-8

Bits do *code point*	Primeiro *code point*	Último *code point*	Bytes	Byte 1	Byte 2	Byte 3	Byte 4
7	U+0000	U+007F	1	0xxxxxxx			
11	U+0080	U+07FF	2	110xxxxx	10xxxxxx		
16	U+0800	U+FFFF	3	1110xxxx	10xxxxxx	10xxxxxx	
21	U+10000	U+1FFFFF	4	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

- 1 byte - Tabela ASCII
- 2 bytes - Caracteres latinos, hebraicos, gregos, ...
- 3 bytes - Caracteres chineses, japoneses, coreanos, ...
- 4 bytes - Alguns outros caracteres e símbolos

Codificação de Caracteres

UTF-8

Representação de strings

C	o	n	c	e	i	ç	ã	o
43	6F	6E	63	65	69	C3A7	C3A3	6F

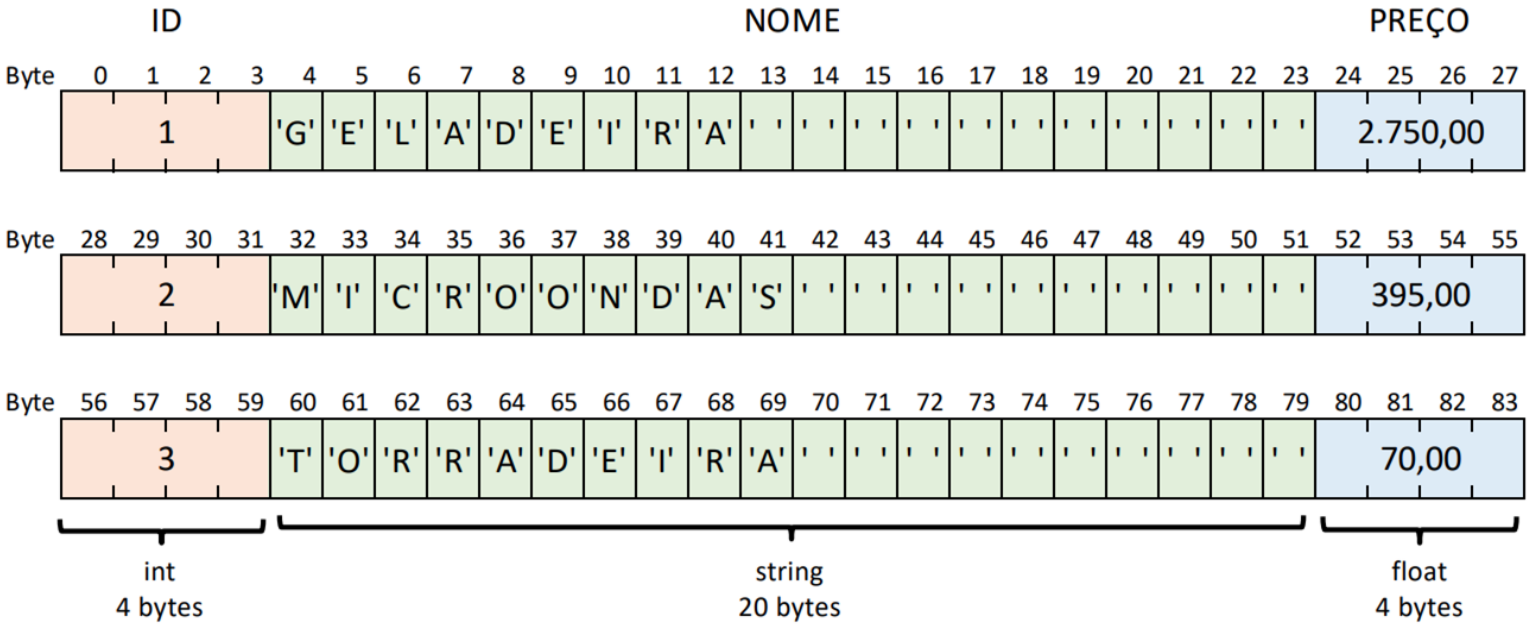
9 caracteres – 11 bytes

https://onlineunicodetools.com/convert-unicode-to-hex

Diferentes Tipos de Registro

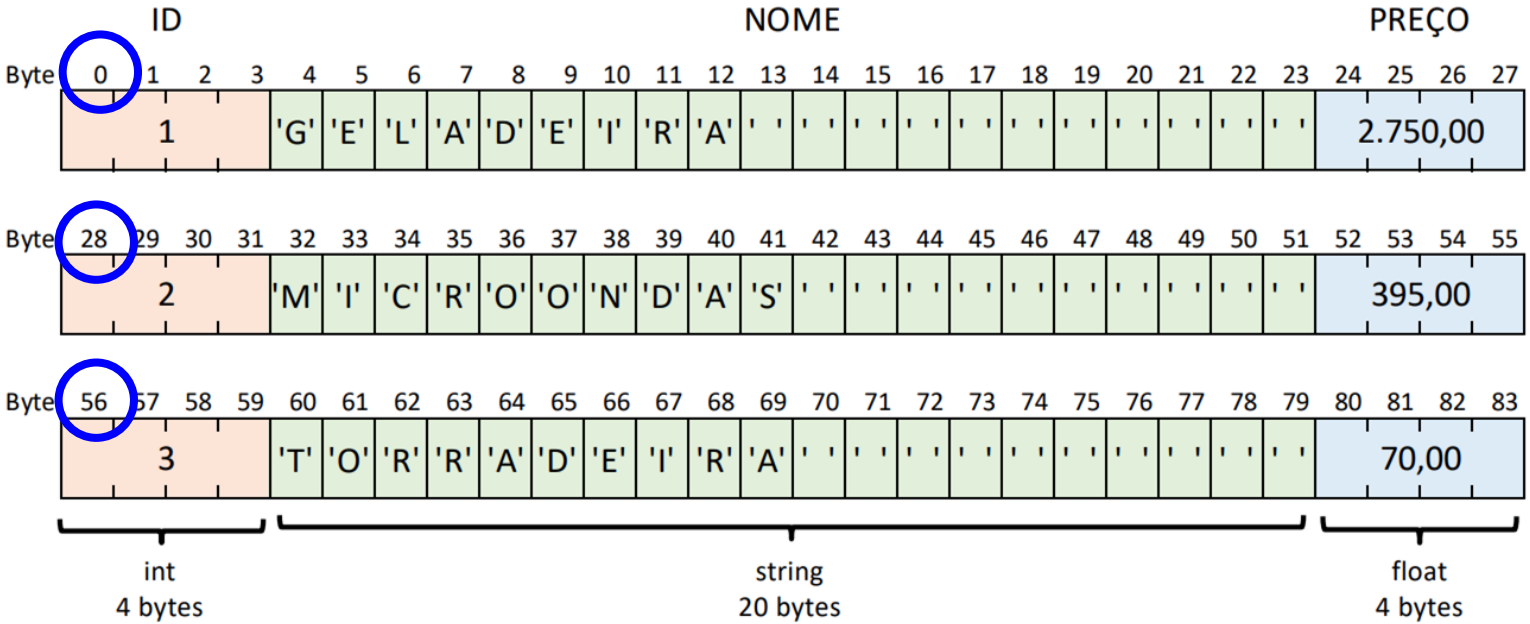
PUC Minas

Registros de tamanho fixo com campos de tamanho fixo



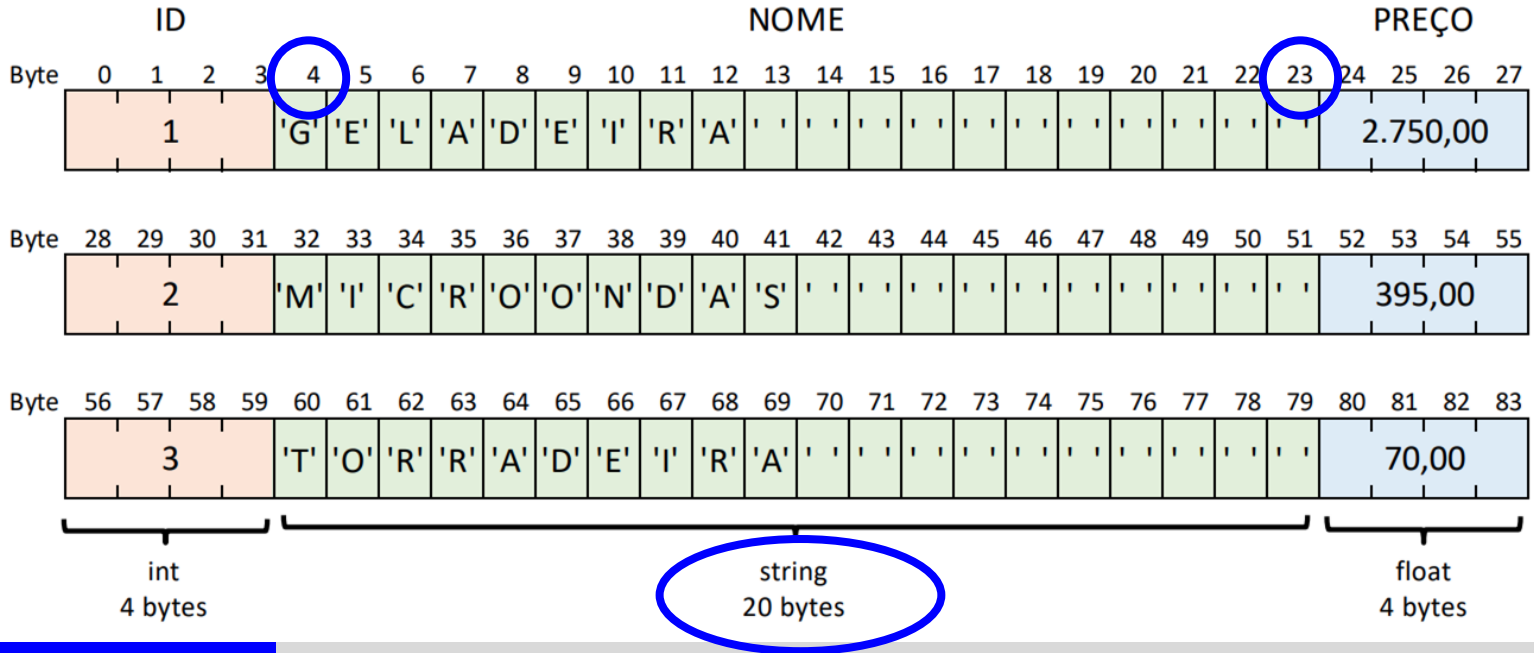
Registros de tamanho fixo com campos de tamanho fixo

Cada entidade tem sempre o mesmo tamanho. Assim, sempre sabemos a posição de início de cada entidade no arquivo.



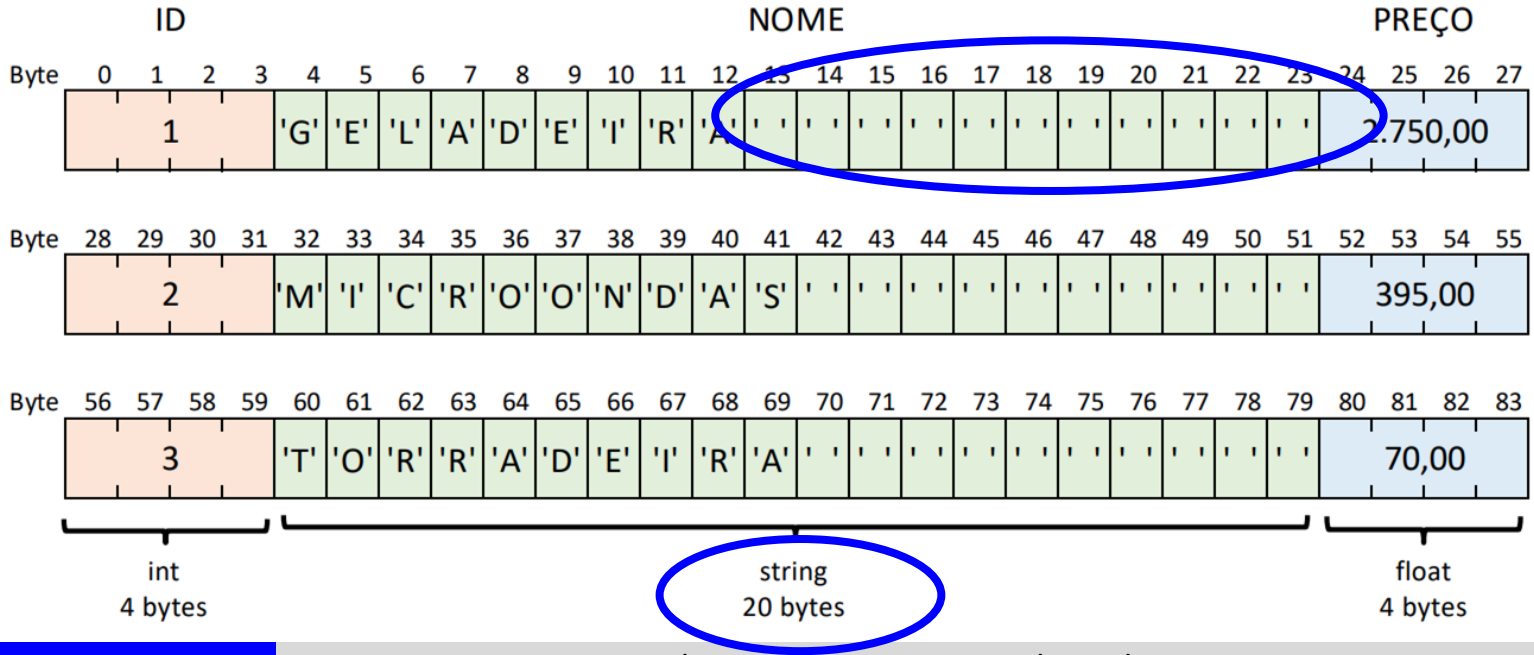
Registros de tamanho fixo com campos de tamanho fixo

Existe um limite de tamanho para as Strings



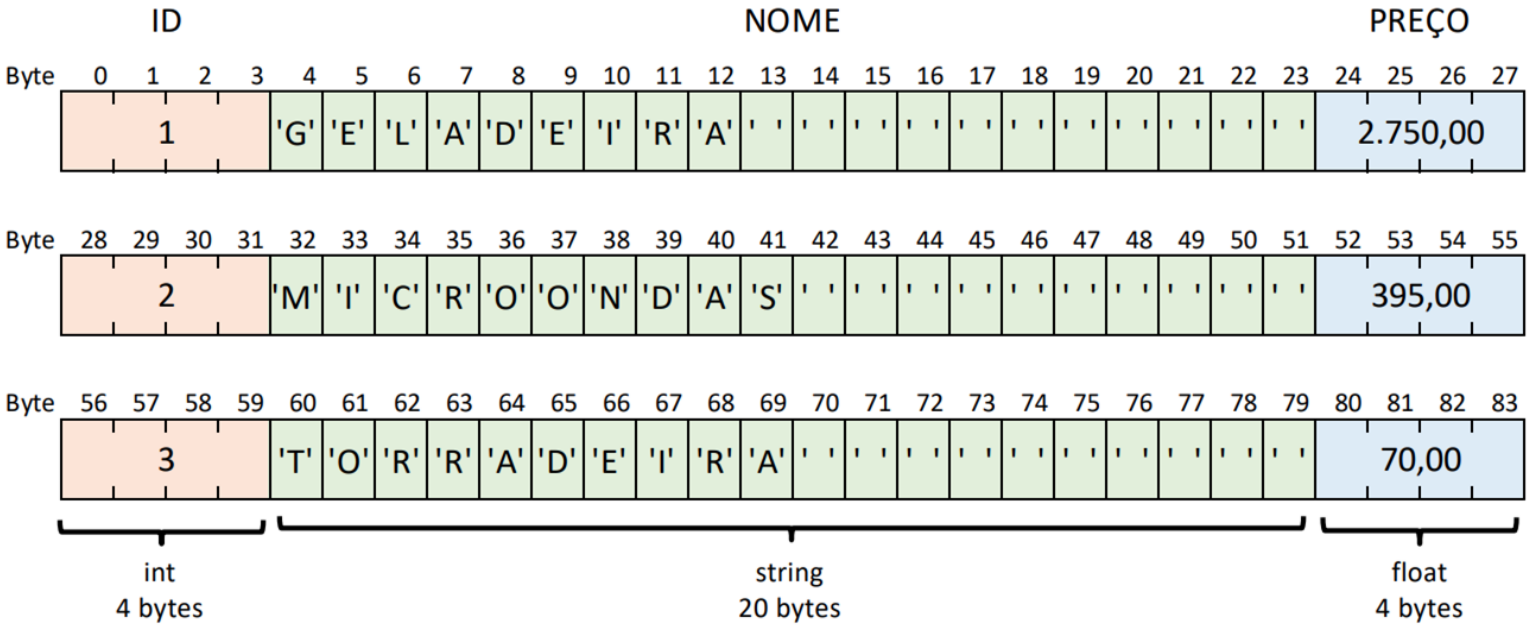
Registros de tamanho fixo com campos de tamanho fixo

Pode haver desperdício ou falta de espaço

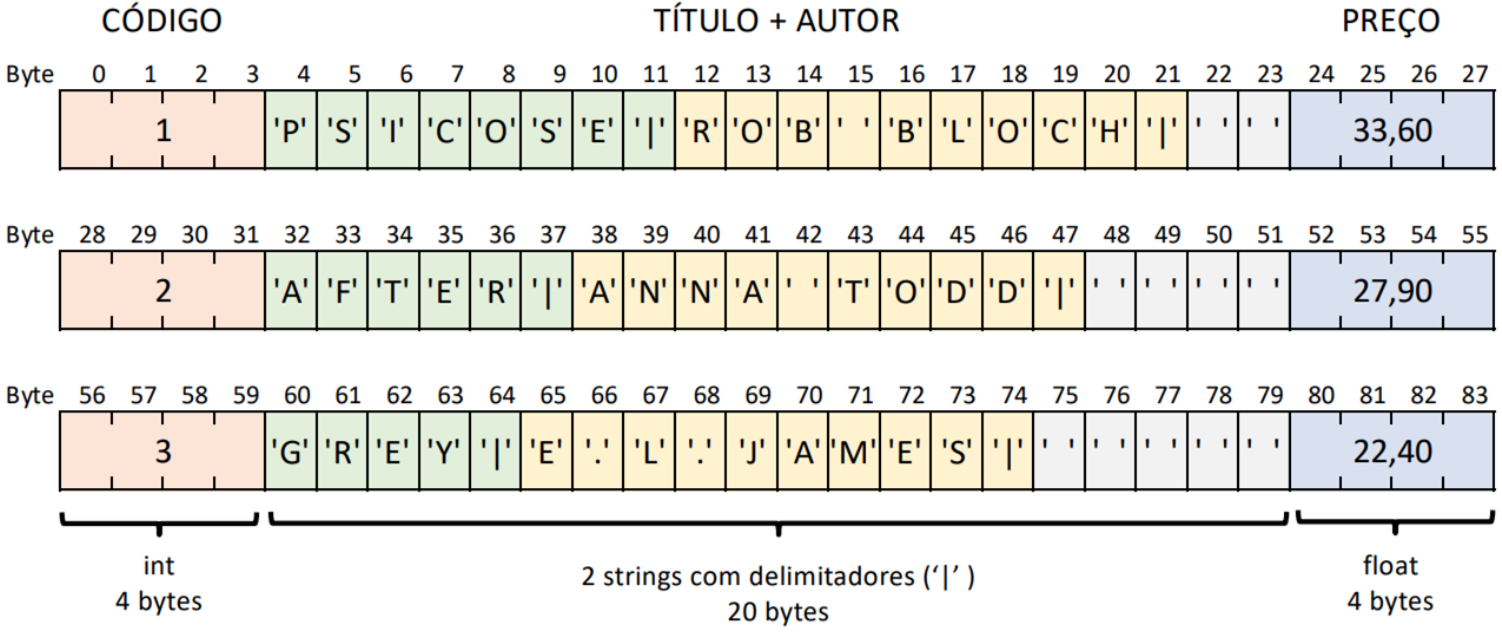


Registros de tamanho fixo com campos de tamanho fixo

É útil para Strings que possuem tamanho fixo. Exemplo: cpf, cnpj, telefone, celular, outros.

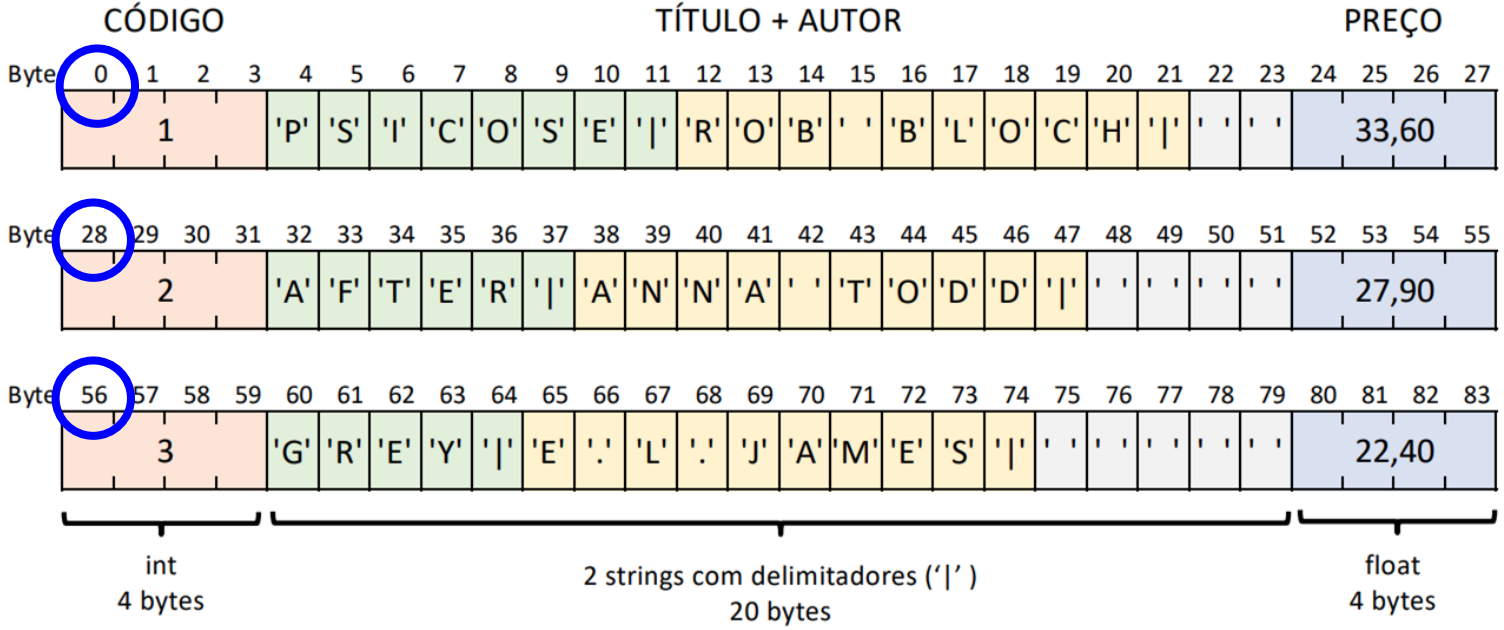


Registros de tamanho fixo com campo de tamanho variável



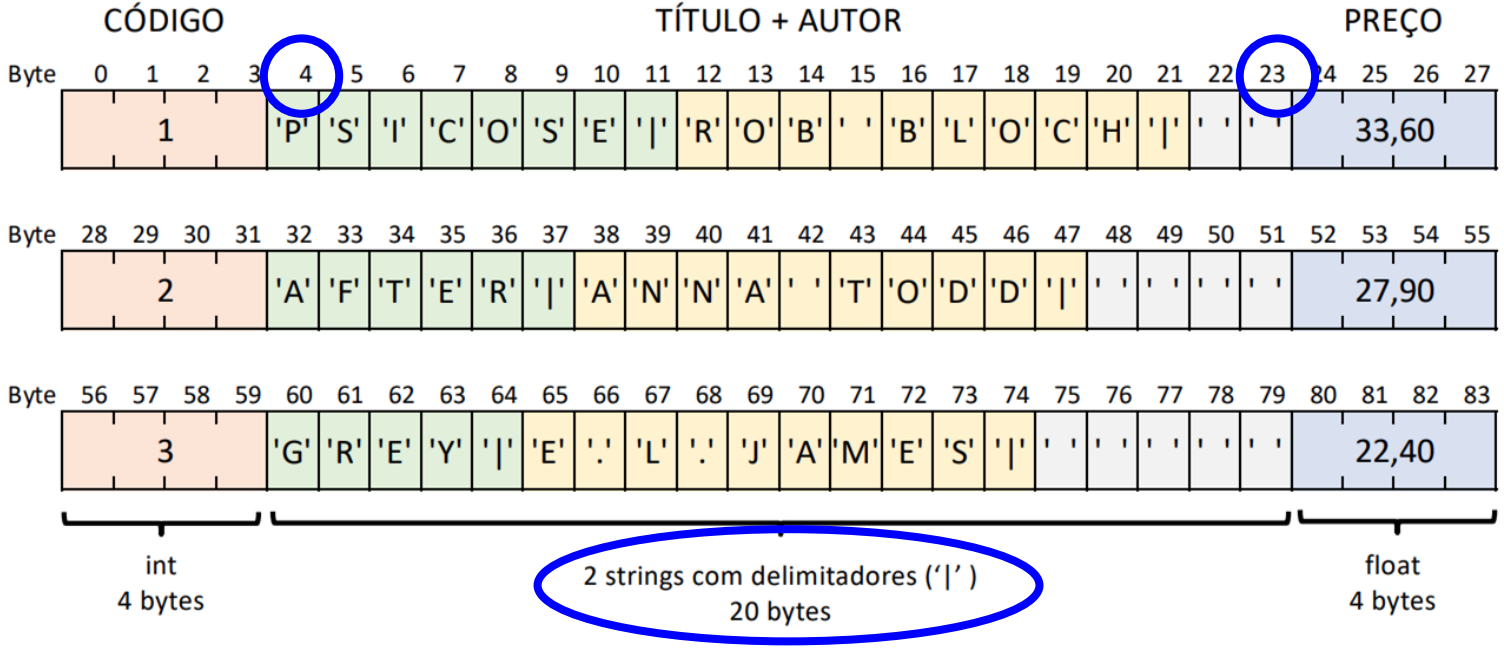
Registros de tamanho fixo com campo de tamanho variável

Cada entidade tem sempre o mesmo tamanho. Assim, sempre sabemos a posição de início de cada entidade no arquivo.



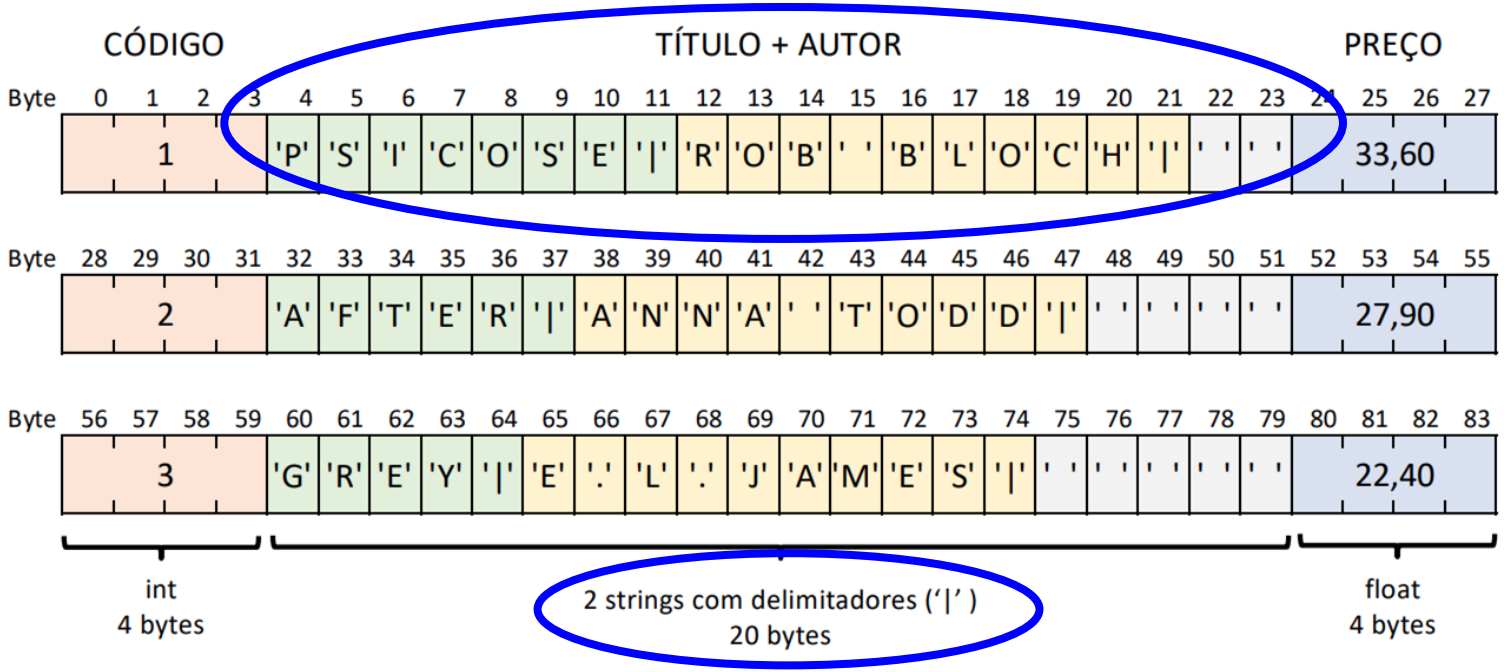
Registros de tamanho fixo com campo de tamanho variável

Existe um limite de tamanho para as Strings



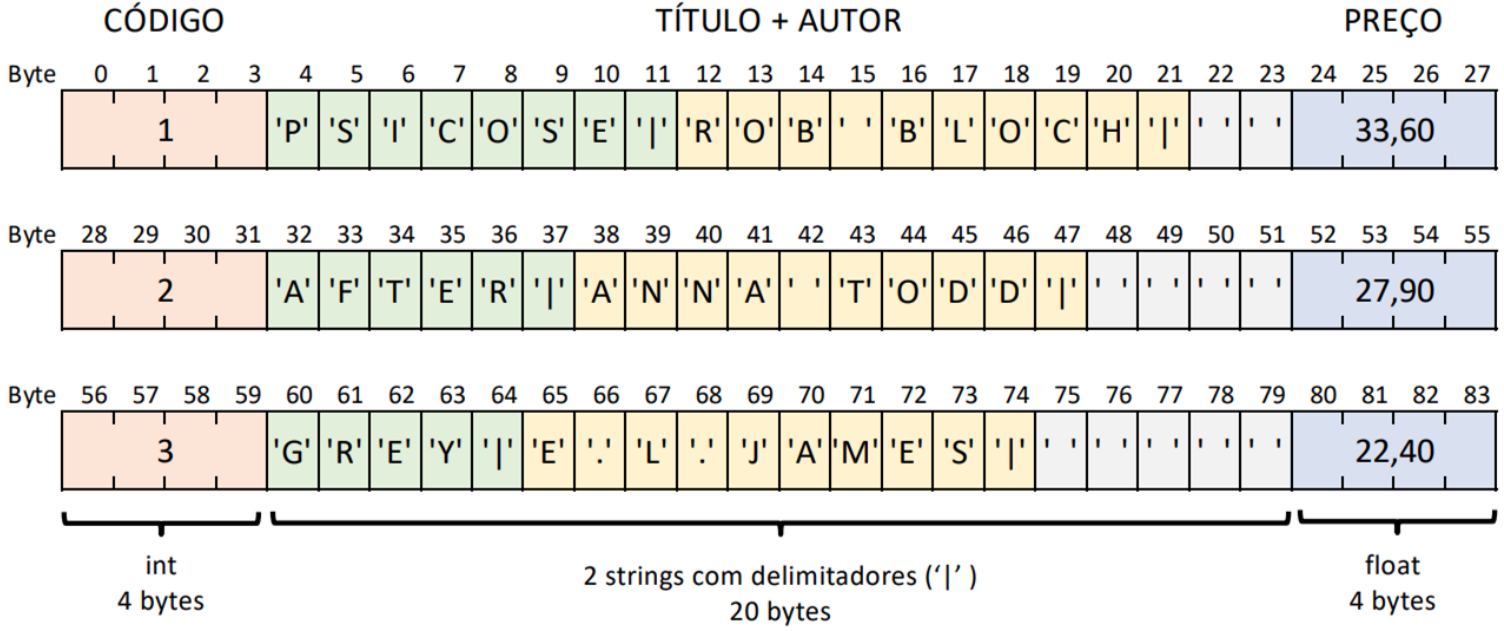
Registros de tamanho fixo com campo de tamanho variável

Pode haver desperdício ou falta de espaço

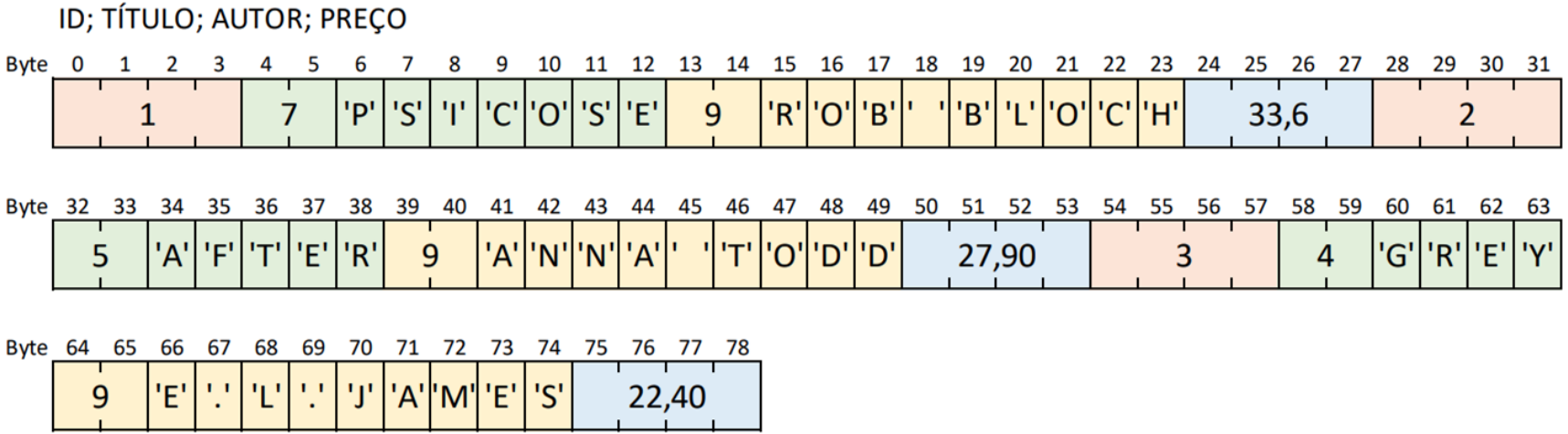


Registros de tamanho fixo com campo de tamanho variável

É útil para Strings que não possuem tamanho fixo, porém existe um limitador de tamanho.
Exemplo: nome + sobrenome, sendo que o máximo aceito são 30 letras.



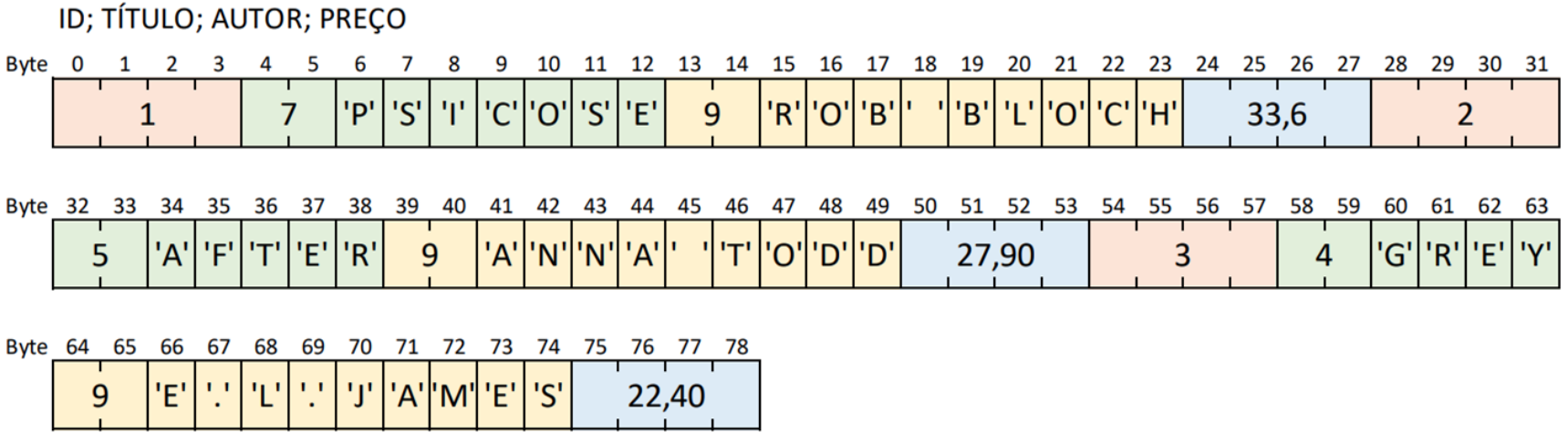
Registros de tamanho variável



Registros de tamanho variável

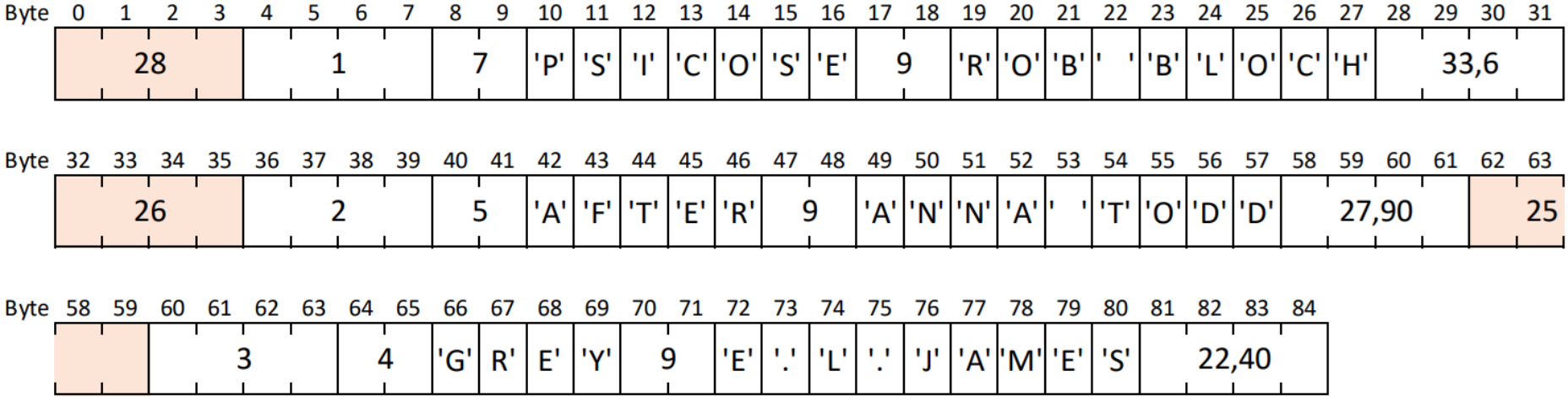
O arquivo se ajusta ao tamanho necessário de cada campo.

Somente o acesso sequencial aos registros é possível.
Porém, o acesso aleatório pode ser feito por meio do uso de índices.



Registros de tamanho variável com indicador de tamanho de registro

ID; TÍTULO; AUTOR; PREÇO



Registros de tamanho variável com indicador de tamanho de registro

O arquivo se ajusta ao tamanho necessário de cada campo.

O indicador de tamanho de registro possibilita “pular” o registro sem precisar percorrê-lo.

ID; TÍTULO; AUTOR; PREÇO

Byte	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
	28				1				7		'P'	'S'	'I'	'C'	'O'	'S'	'E'	9		'R'	'O'	'B'	' '	'B'	'L'	'O'	'C'	'H'	33,6			

Byte	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
	26				2				5		'A'	'F'	'T'	'E'	'R'	9		'A'	'N'	'N'	'A'	' '	'T'	'O'	'D'	'D'	27,90				25	

Byte	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84
			3				4		'G'	'R'	'E'	'Y'	9		'E'	'.'	'L'	'.'	'J'	'A'	'M'	'E'	'S'	22,40			

Qual o melhor tipo de registro?

Depende do contexto e do problema que está sendo resolvido.

O objetivo é obter:

- o **menor número de operações**
  - CRUD em memória secundária tem alto custo.
- e também o **menor tamanho de arquivo.**

Roteiro da Aula

- Manipulando a Memória secundária
  - Arquivos
  - Arquivos como vetores de bytes
  - Fluxos de entrada e saída
  - **CRUD (Implementando em JAVA)**

Serialização

Conversão de um objeto complexo em uma sequência de tipos primitivos

• As linguagens de programação já oferecem métodos para escrever os tipos primitivos (+ string) como uma sequência de bytes

Entidade:

```
Cliente {
  short    ID;
  String   nome;
  String[] emails;
}
```

Registro:

```
ID              → short de 2 bytes
nome            → string UTF-8 com indicador de tamanho
emails.length   → byte para indicar quantidade de emails
emails[0]       → string UTF-8 com indicador de tamanho
emails[1]       → string UTF-8 com indicador de tamanho
...
emails[n-1]     → string UTF-8 com indicador de tamanho
```

Serialização

Conversão de um

• As linguagens de (+ string) como uma s

Dúvida???

Como organizar uma quantidade (que eu não sei qual é) de emails em um arquivo?

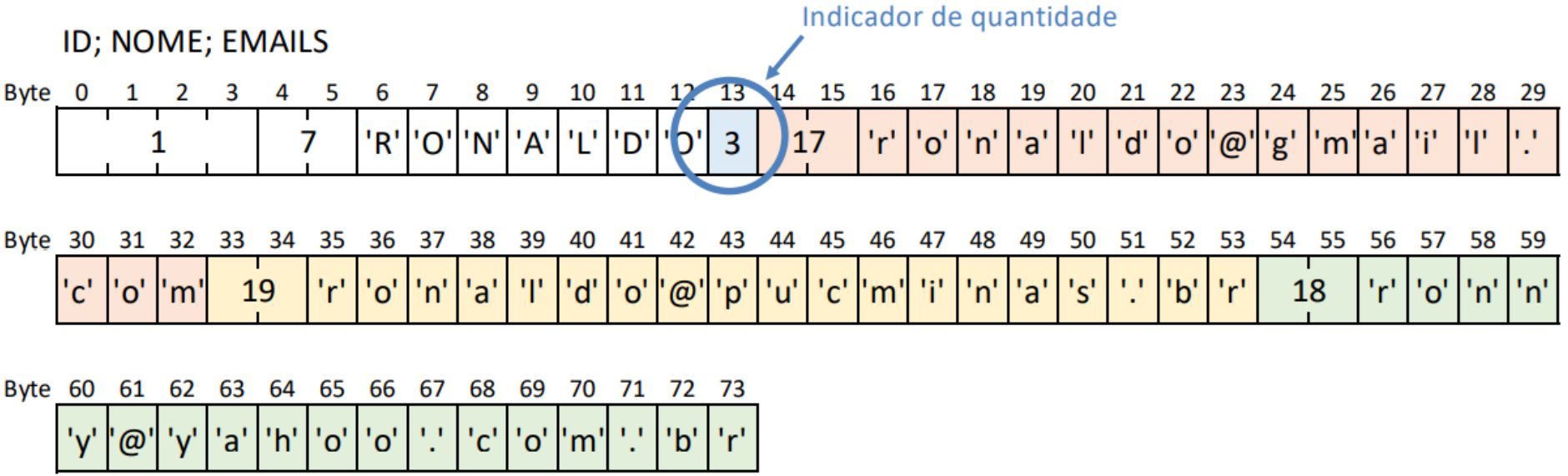
Entidade:

```
Cliente {
  short    ID;
  String   nome;
  String[] emails;
}
```

Registro:

```
ID             → short de 2 bytes
nome           → string UTF-8 com indicador de tamanho
emails.length  → byte para indicar quantidade de emails
emails[0]      → string UTF-8 com indicador de tamanho
emails[1]      → string UTF-8 com indicador de tamanho
...
emails[n-1]    → string UTF-8 com indicador de tamanho
```

- Exemplo – emails de um cliente:



FileOutputStream e DataOutputStream

- A classe FileOutputStream nos oferece apenas o método write() para escrevermos dados no arquivo.
- Assim, precisamos do apoio de uma segunda classe que nos oferece métodos para a escrita dos tipos primitivos (int, float, boolean, etc.). Essa classe é a DataOutputStream.
    - Ela contém métodos como writeInt() para a escrita de números inteiros, writeFloat() para a escrita de números reais, writeUTF() para a escrita de strings e muito mais.

FileInputStream e DataInputStream

- A classe FileInputStream nos oferece apenas o método read() para lermos dados no arquivo.
- Assim, precisamos do apoio de uma segunda classe que nos oferece métodos para a leitura dos tipos primitivos (int, float, boolean, etc.). Essa classe é a DataInputStream.
  - Ela contém métodos como readInt() para a escrita de números inteiros, readFloat() para a escrita de números reais, readUTF() para a escrita de strings e muito mais.

ByteArrayOutputStream e ByteArrayInputStream

- Essas duas classes funcionam de forma parecida a FileOutputStream e FileInputStream, mas ao invés de fazerem escrita e leitura para arquivos, fazem essas operações para memória - usando também a ideia de fluxos de entrada e de saída.
- O registro de cada objeto será precedido por um indicador de tamanho.
- Dessa forma, o programa principal não precisa entender que informações o registro contém - apenas precisa ser capaz de ler e escrever vetores de bytes de tamanho variável (os registros).

RandomAccessFile

- Classe RandomAccessFile usada para fazermos leitura e escrita
- Há apenas um ponteiro que indica qual será a próxima posição a ser lida ou escrita. Esse ponteiro pode ser identificado por meio do método getFilePointer() e pode ser movimentado por meio do método seek().

Observações finais

Os códigos serão disponibilizados no Canvas.