Unidade II - Tipos, Operadores e Expressões

Disciplina Linguagens de Programação I
Bacharelado em Ciência da Computação da Uerj
Professores Guilherme Mota e Leandro Marzulo

ANSI C

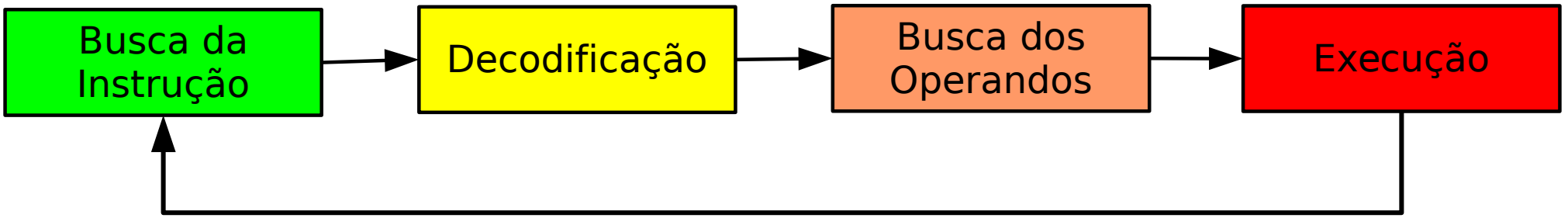
```c
#include <stdio.h>
int main ()
{
   printf("Hello World!");
   return 0;
}
```

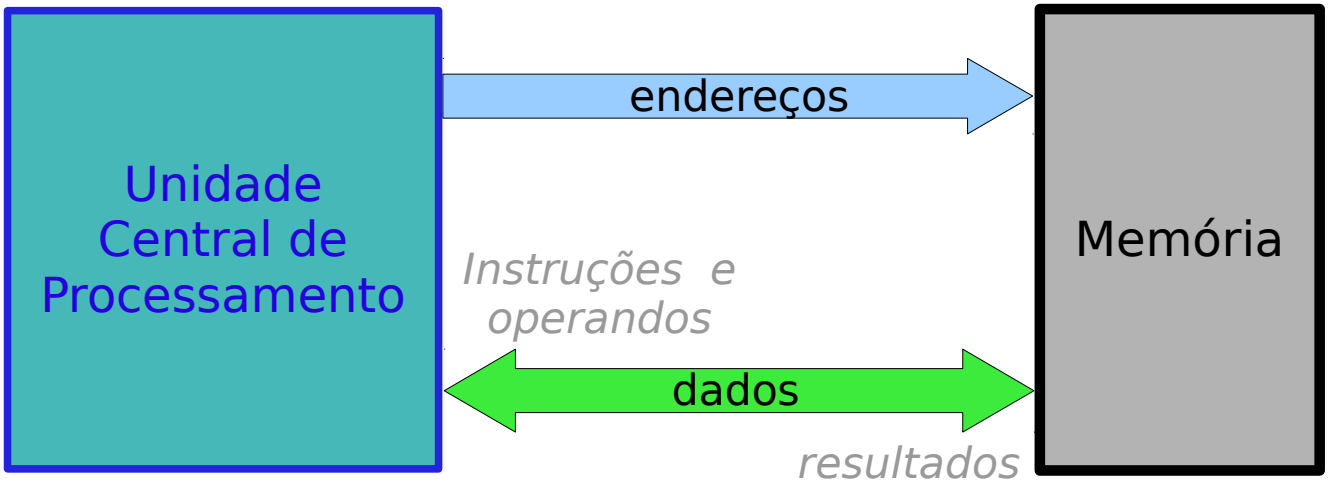
Que assuntos serão abordados nesta unidade?

- Organização de Computadores
  - visão geral
  - modelo em níveis
  - subsistema de memória
  - uso dos sistemas de tipos
- Operadores e Expressões no ANSI C
  - operadores
  - precedência de operadores
  - ordem de avaliação
  - conversão de tipos
  - operações com strings
- Sistema de tipos do ANSI C
  - tipos primários
  - declaração de constantes
  - tipos derivados
  - tipos definidos no programa
  - declaração de variáveis
  - alocação de memória
    - alocação automática
    - alocação dinâmica
  - variáveis em registradores

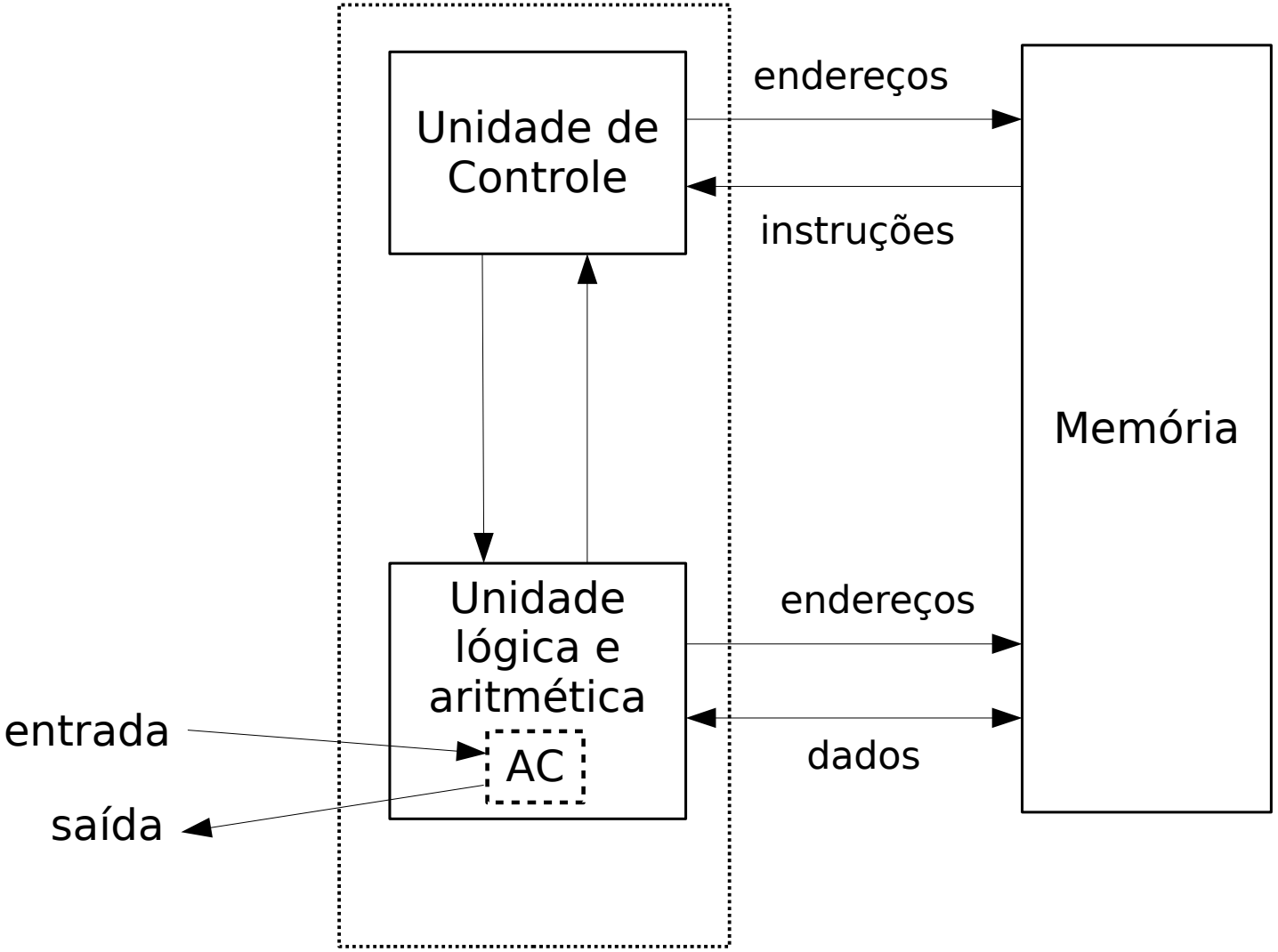
Estrutura básica de um computador

Ciclo de Instrução

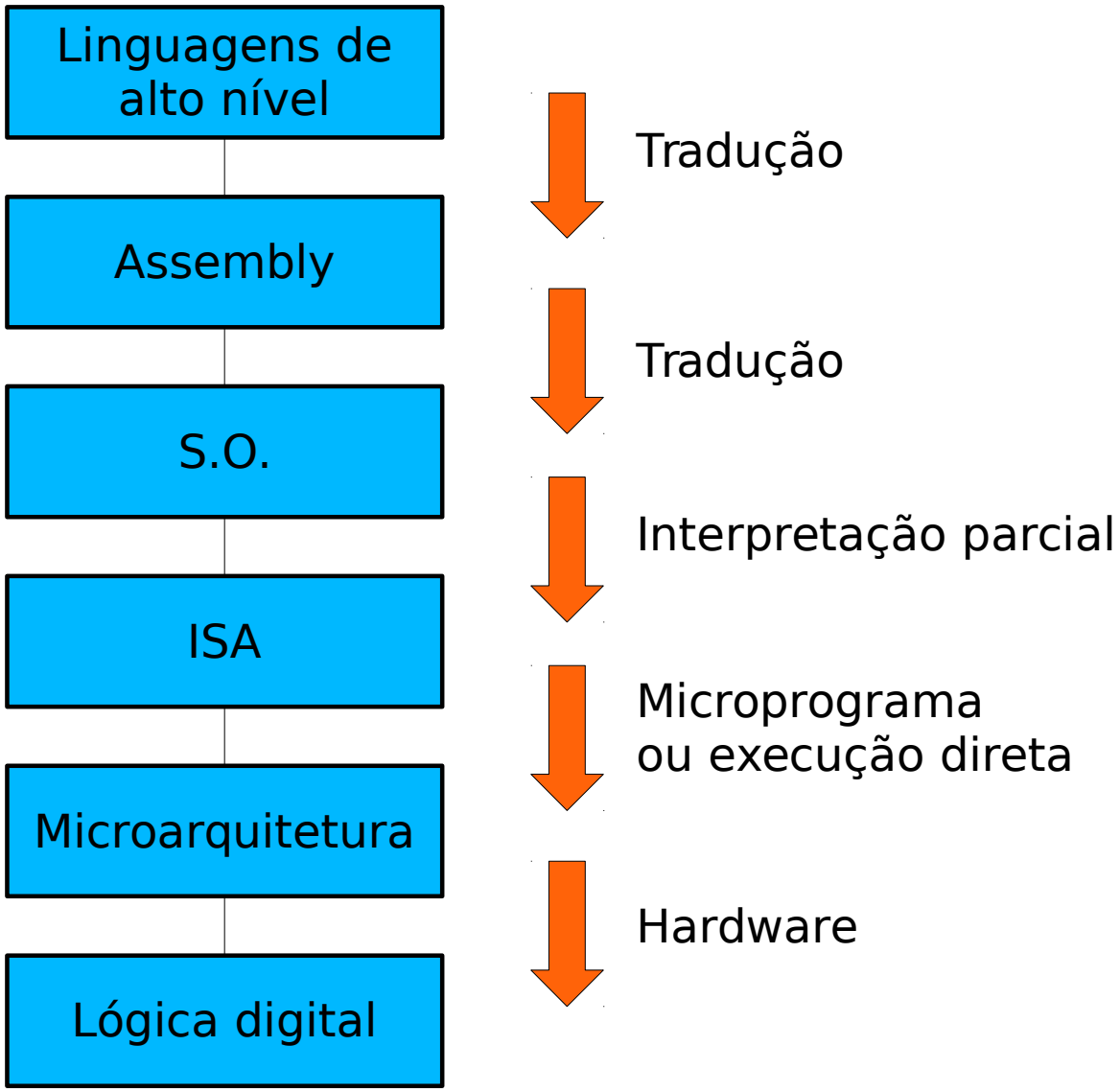




A Máquina Original de Von Neuman

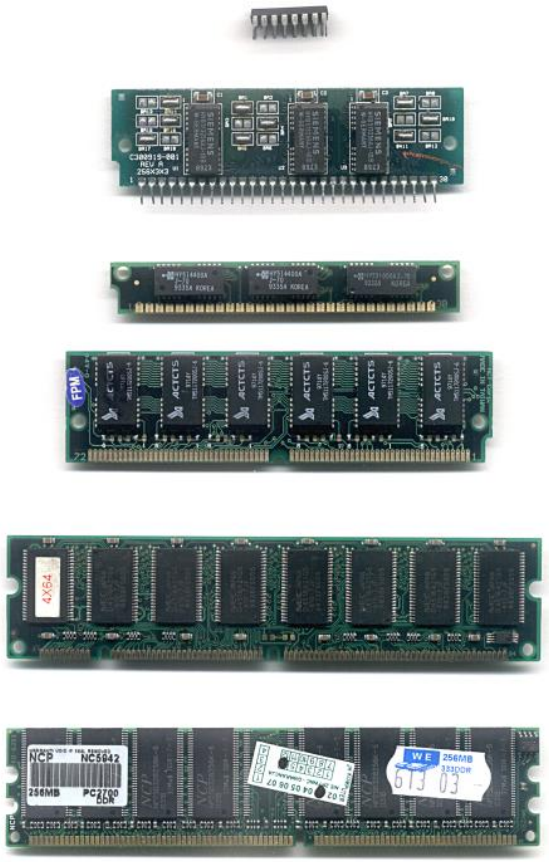


Arquitetura de Computadores em Níveis

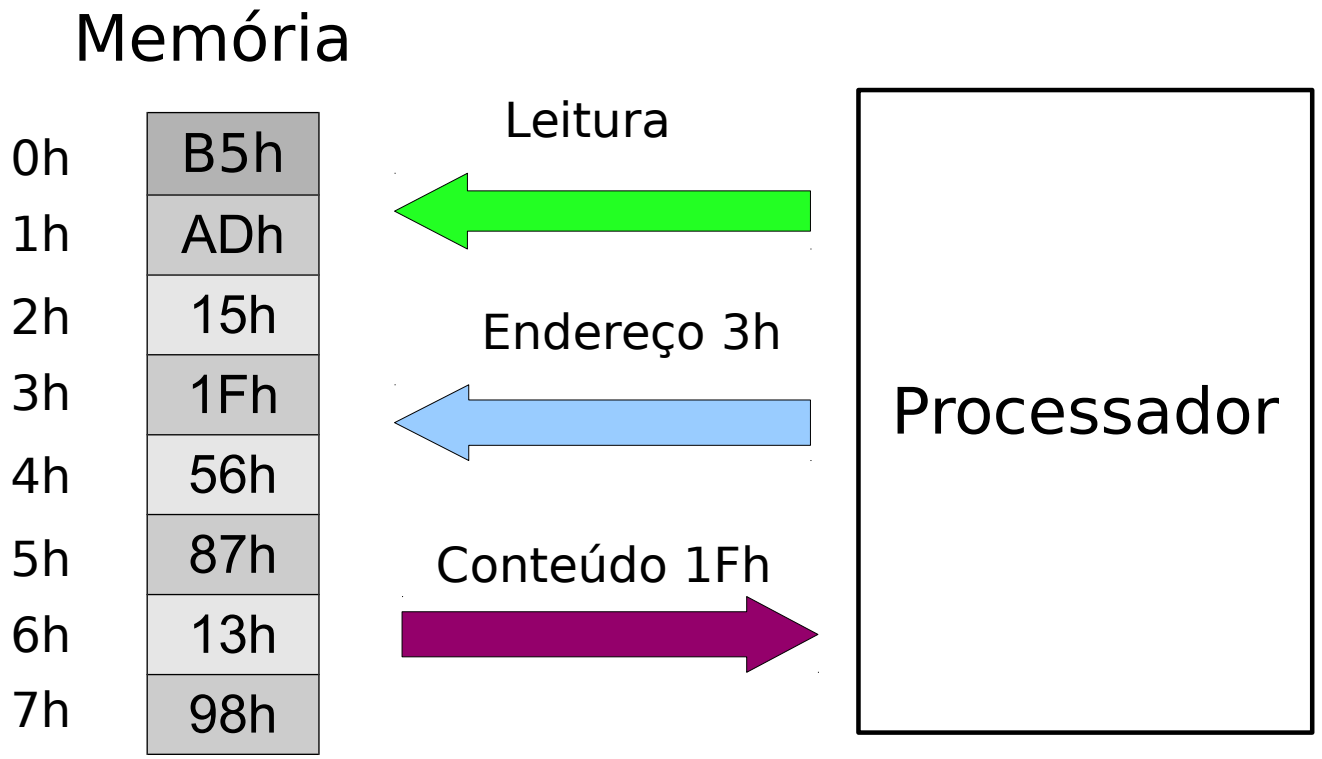


Memória de computador

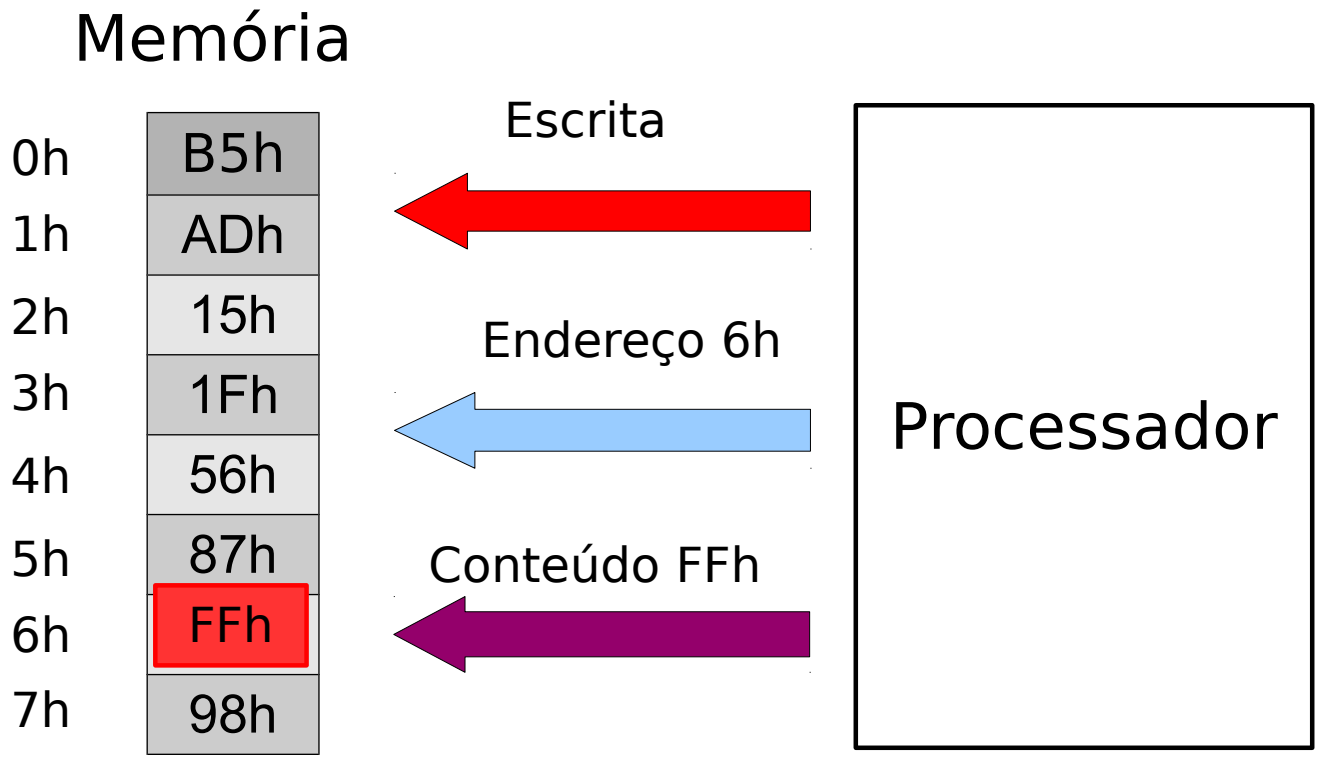
Dispositivos de Memória



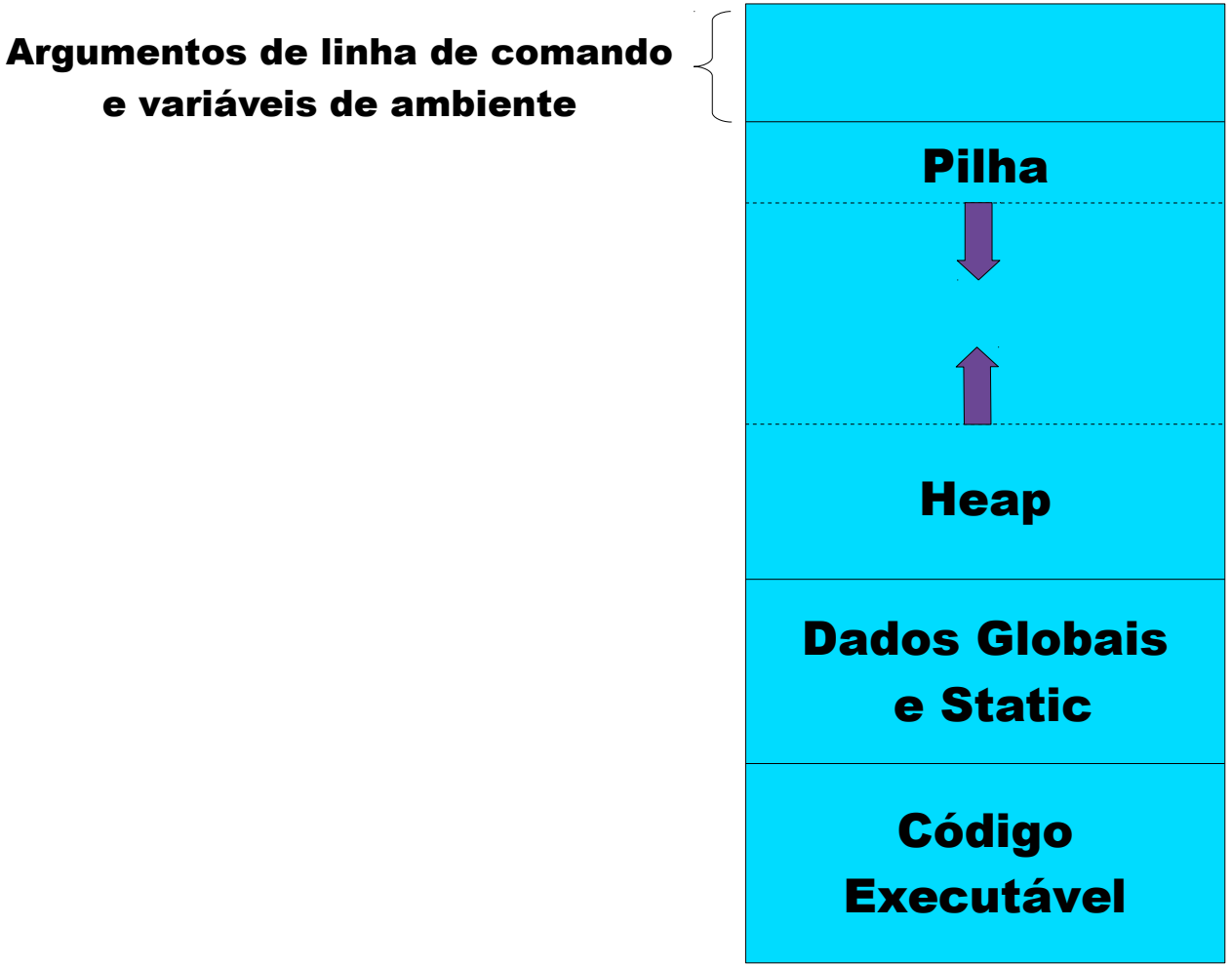
Leitura da Memoria



Escrita na Memoria



Mapa de Memória de um Programa C em Execução



Sistema de tipos

Funções dos Sistemas de Tipos

- Armazenamento de dados em memória
  - Codificação
  - Decodificação
- Execução de expressões
  - Precedência
  - Ordem de avaliação
- Conversão de valores
- Validação de tipos

Sistema de Tipos ANSI C

- Tipos primários
  - `char, short, int, long, long long, (signed / unsigned)`
  - `float, double, long double`
- Tipos derivados
  - `Tipo [], Tipo [][], Tipo *, Tipo**, struct {}, union`
- Tipos definidos pelo usuário
  - `enum,typedef`

Tipos primários do ANSI C

Tipos Primários do Ansi C

Tipo	Tam. (B)	Faixa de valores *signed*	Faixa de valores *unsigned*
`char`	1	[-128, 127]	[0, 255]
`short`	2	[-32.768, 32.767]	[0, 65.535]
`int`	4	[-2.147.483.648, 2.147.483.647]	[0, 4.294.967.295]
`long`	4	[-2.147.483.648, 2.147.483.647]	[0, 4.294.967.295]
`long long`	8	[-9.223.372.036.854.775.808, 9.223.372.036.854.775.807]	[0, 18.446.744.073.709.551.615]

Tipos Primários do Ansi C

- A codificação dos números reais se assemelha à notação científica. Ex.: $678 \rightarrow 6{,}78 \times 10^2$

S	EXPOENTE	MANTISSA

$$Valor = (-1)^S . M . 2^E$$

Tipo	Tam. (B)	Faixa de valores*
float	4	$[3{,}4 \times 10^{-38}, 3{,}4 \times 10^{38}]$
double	8	$[1{,}7 \times 10^{-308}, 1{,}7 \times 10^{308}]$
long double	10	$[3{,}4 \times 10^{-4932}, 3{,}4 \times 10^{4932}]$

* fonte: “Programming in Ansi C” E. Balagurusamy

Definição de constantes

Constantes

```
'A' 'c'   /* (char) */
"Cadeia de char" "" /* (char[]) */
100   /* (int) */
100L /* (long) */
100.0   /* (float) */
123.7e-2  123.7E-2   /* (double) */
023 067L   /* (int) notação octal */
0xFD 0XFD 0xdaefeeL    /* (int) notação hex */
'\n' '\t' '\b' '\0' '\'' '\\' '\014'/* (char) */
```

Código ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Código ASCII

128	Ç	144	É	160	á	176	▒	192	└	208	╨	224	α	240	≡
129	ü	145	æ	161	í	177	▒	193	┴	209	╤	225	ß	241	±
130	é	146	Æ	162	ó	178	▓	194	┬	210	╥	226	Γ	242	≥
131	â	147	ô	163	ú	179	│	195	├	211	╙	227	π	243	≤
132	ä	148	ö	164	ñ	180	┤	196	─	212	╘	228	Σ	244	⌠
133	à	149	ò	165	Ñ	181	╡	197	┼	213	╒	229	σ	245	⌡
134	å	150	û	166	ª	182	╢	198	╞	214	╓	230	µ	246	÷
135	ç	151	ù	167	º	183	╖	199	╟	215	╫	231	τ	247	≈
136	ê	152	ÿ	168	¿	184	╕	200	╚	216	╪	232	Φ	248	°
137	ë	153	Ö	169	⌐	185	╣	201	╔	217	┘	233	Θ	249	∙
138	è	154	Ü	170	¬	186	║	202	╩	218	┌	234	Ω	250	·
139	ï	155	¢	171	½	187	╗	203	╦	219	█	235	δ	251	√
140	î	156	£	172	¼	188	╝	204	╠	220	▄	236	∞	252	ⁿ
141	ì	157	¥	173	¡	189	╜	205	═	221	▌	237	φ	253	²
142	Ä	158	₧	174	«	190	╛	206	╬	222	▐	238	ε	254	■
143	Å	159	ƒ	175	»	191	┐	207	╧	223	▀	239	∩	255	

Source: www.LookupTables.com

Exercício U2.1 - Tipos primários no Linux 64

Crie um programa na linguagem C que apresente na tela os tamanhos dos tipos primários da linguagem (char, short, int, long, long long, float, double e long double).

Tipos derivados do ANSI C

Arrays

`float[]`

0.27	2.45	6.12	5.07	0.145

`int[][]`

10	20	15	18
44	65	2	5
16	25	6	36
23	15	56	36

struct

- Cria tipos compostos

```
struct <identificadorDoTipo>
{
   <tipo1> <identificadorMembro1>;
   ...
   <tipon> <identificadorMembron>;
};
```

```
struct Point2D
{
   float x;
   float y;
   int label;
};
```

struct

```
struct Point2D
{
   float x;
   float y;
   int label;
} Var1;

struct Point2D P1;

P1.x = 10.0;
P1.y = 20.0;
P1.label = 20;

Var1 = P1;
```

union

- Cria tipos que pode assumir valores e se comportar como tipos diferentes
- Não é feito teste de consistência, o programa deve saber o tipo em uso

```
union u_tag{
   int ival;
   float fval;
   char *sval;
} u;

printf("%d\n", u.ival);
printf("%f\n", u.fval);
printf("%s\n", u.sval);
```

Apontadores

```
float* pfloat;
```

```
float** ppFloat;
```

Tipos definidos no programa

typedef

- Cria novos tipos
- Sinônimo para um tipo primitivo ou derivado

```
typedef <tipo> <identificadorDoTipo>;
```

```
typedef int Idade;
typedef char* String;
```

enum

- Cria um tipos que enumera constantes simbólicas do tipo inteiro
- Não é feito teste de consistência entre o valor inteiro armazenado e as constantes listadas

```
enum booleano { NO, YES };/* NO = 0, YES = 1 */

enum escapes { BELL = '\a',
   BACKSPACE = '\b', TAB = '\t',
   NEWLINE = '\n', VTAB = '\v', RETURN='\r' };

enum months { JAN = 1, FEB, MAR, APR, MAY,
JUN,JUL, AUG, SEP, OCT, NOV, DEC }; /* FEB = 2,
MAR = 3, etc. */
```

Exercício U2.2 - Tipos derivados no Linux 64

1. Baixe o arquivo Ud3Exercicio2.c

2. Analise o código

3. Compile e execute este programa

4. Compare o tamanho dos tipos derivados com o dos respectivos tipos primitivos

5. Avalie a coerência dos tamanhos dos tipos em função da teoria vista na aula

Declaração de Variáveis

Declaração de Variáveis

```
<tipo> <identificadorDaVariavel>;
```

```
int fahr;
char c;
unsigned int indice;
long double saldoContaEikeBatista;
short cont;
float salarioDeProfessor;
float vector[3];
```

Declaração e Inicialização de Variáveis

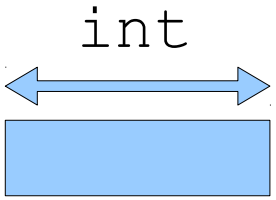
```
<tipo> <Variavel> = <constante>;
```

```
int fahr = 50;
char c = 50;
char c = '2';
char c = 0x32;
char c = 062;
unsigned int indice = 50;
long double saldoContaEikeBatista = 50.7e127;
short cont = 50;
float salarioDeProfessor = 50;
float vector[3] = {50, 12, 8};
char str1[] = "Sou uma cadeia de carcteres\n";
```

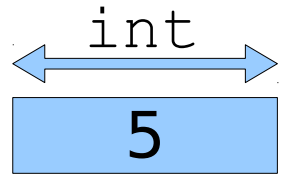
Alocação automática e dinâmica de memória

Alocação automática de memória

```
int number;
```

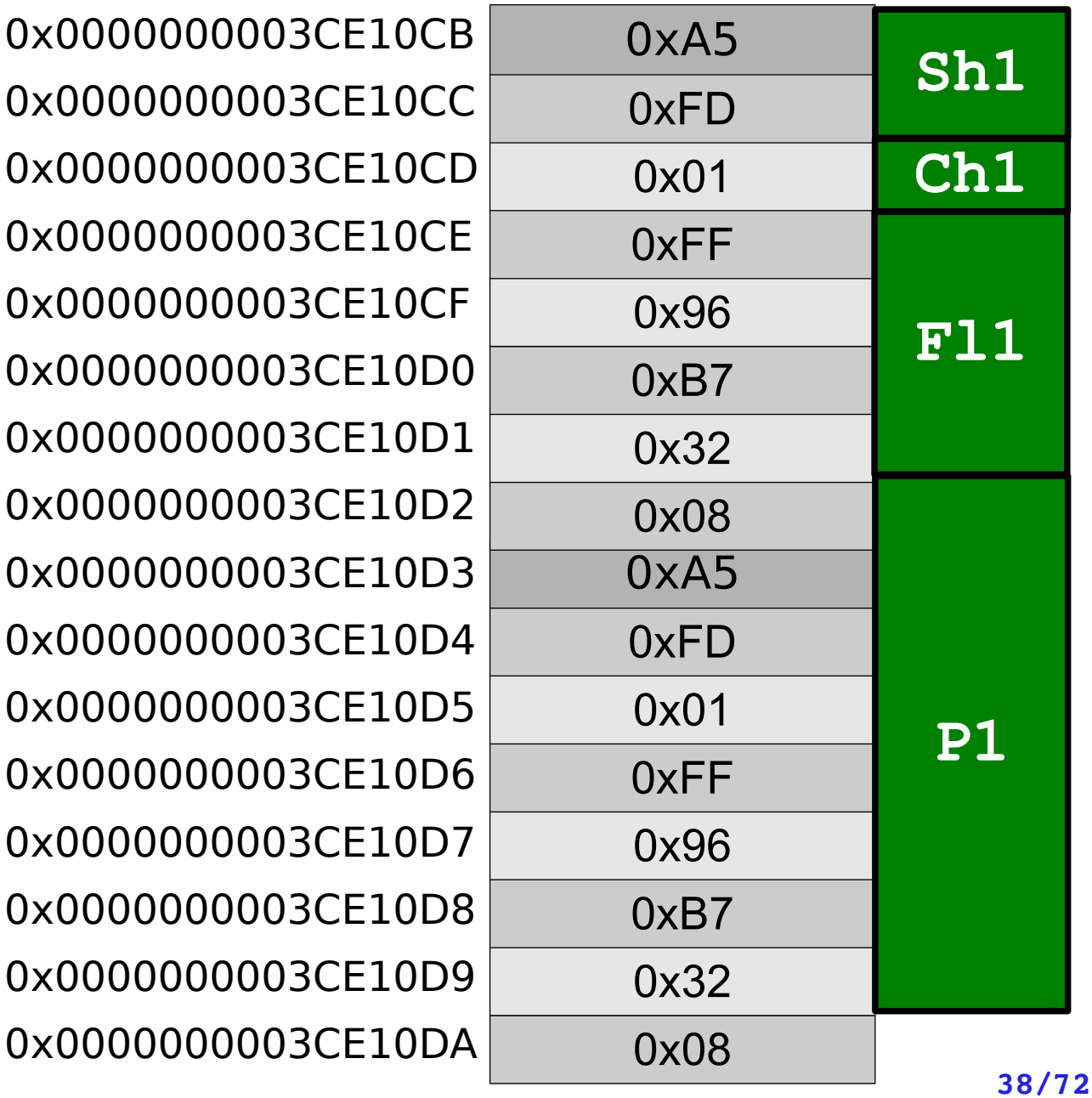



```
int number = 5;
```



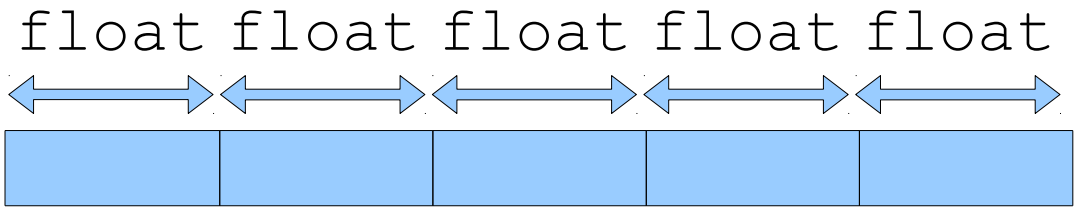
Alocação automática de memória

```
int main()
{
   short Sh1;
   char Ch1;
   float Fl1;
   struct Point2d
   {
    float x,y;
   }P1;
}
```

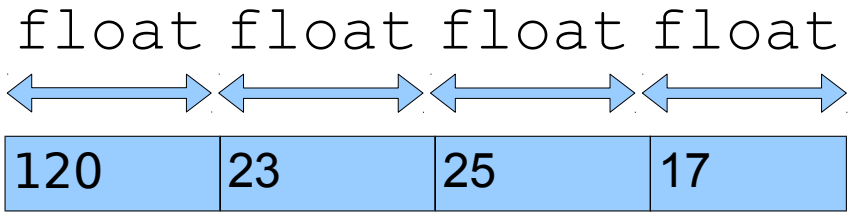


Alocação automática de memória

```
float vect[5];
```

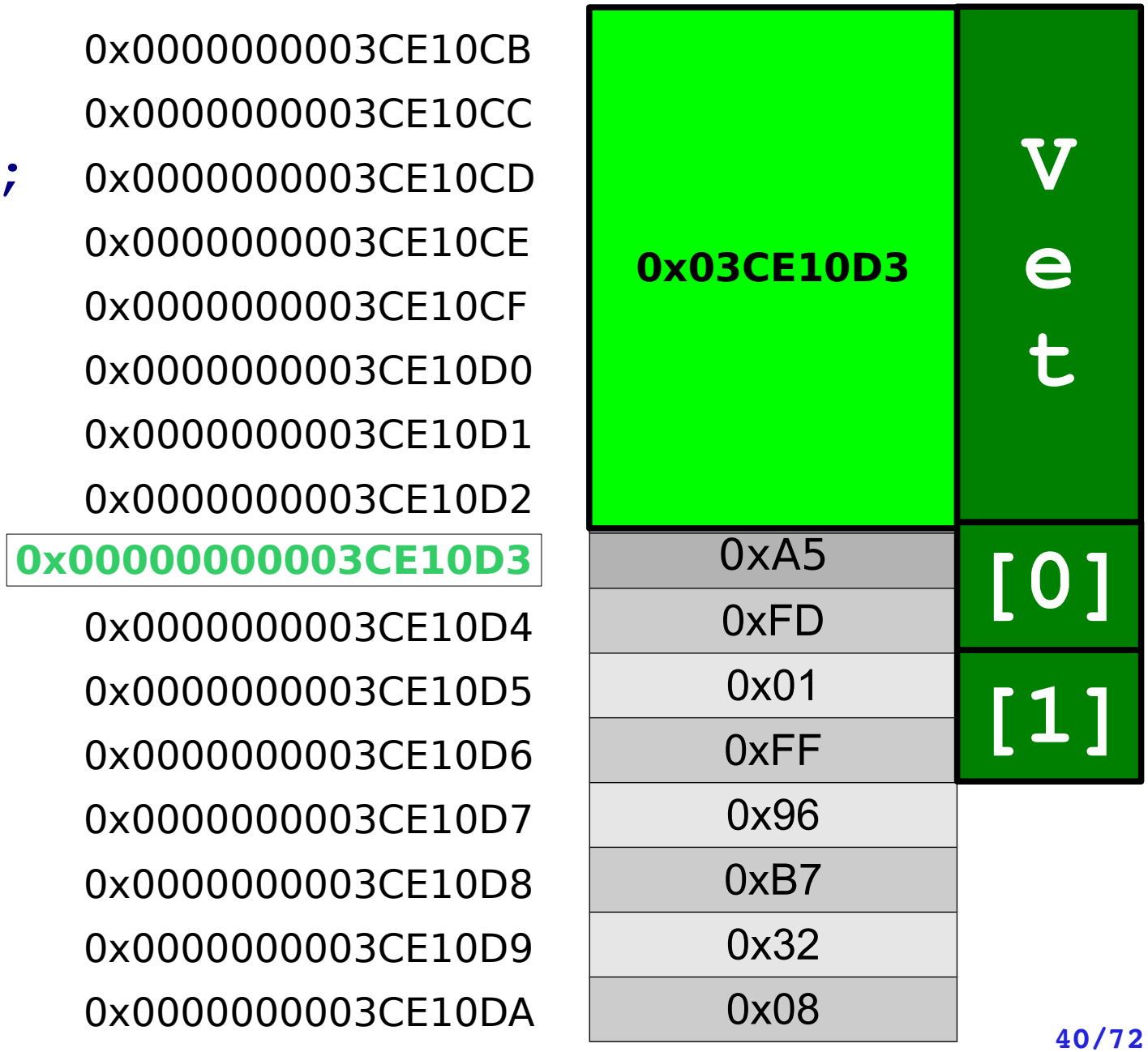



```
float vect2[]={120.0, 23.0, 25.0, 17.0};
```



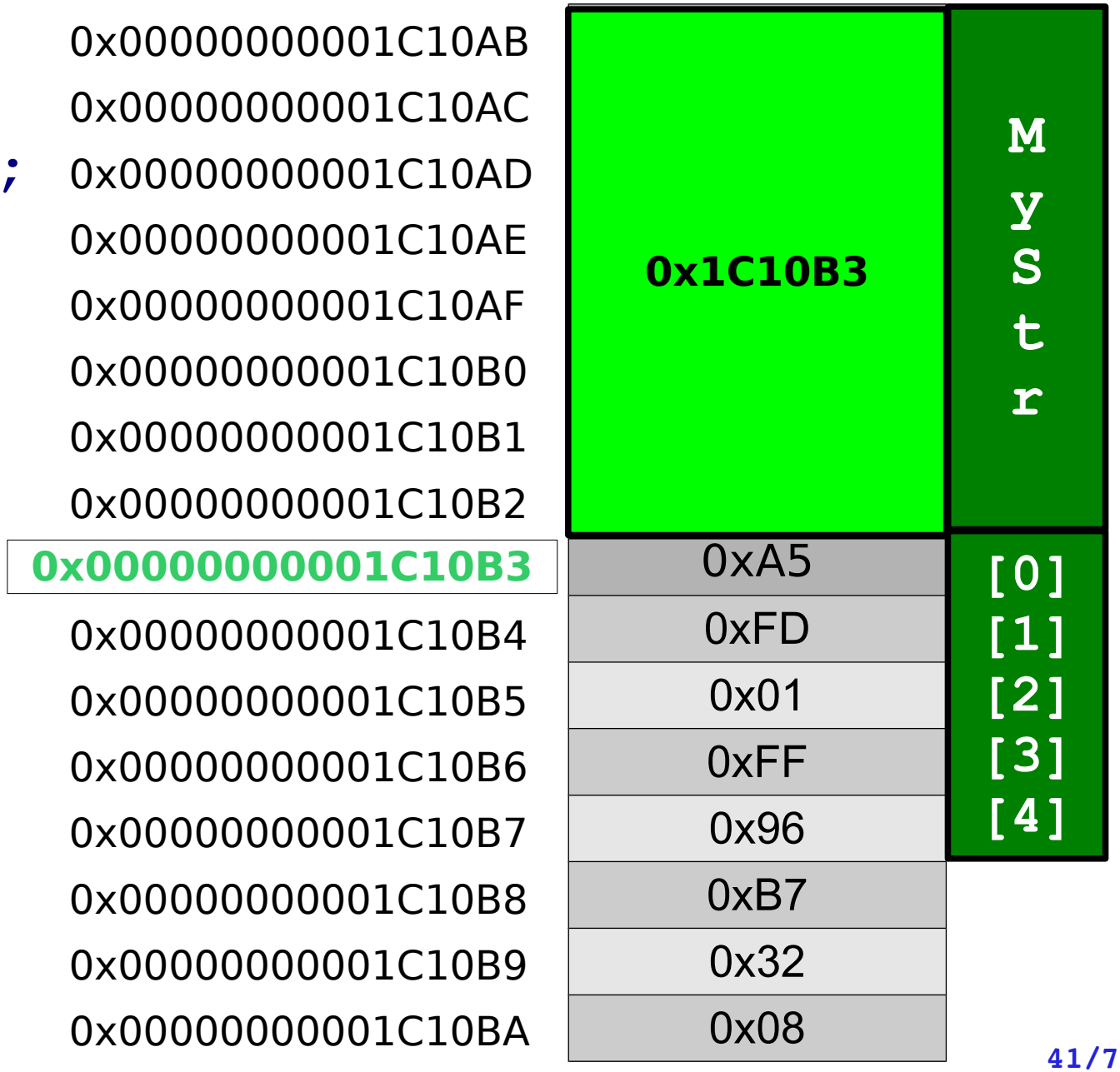
Alocação automática de memória

```
int main()
{
   short Vet[2];
}
```



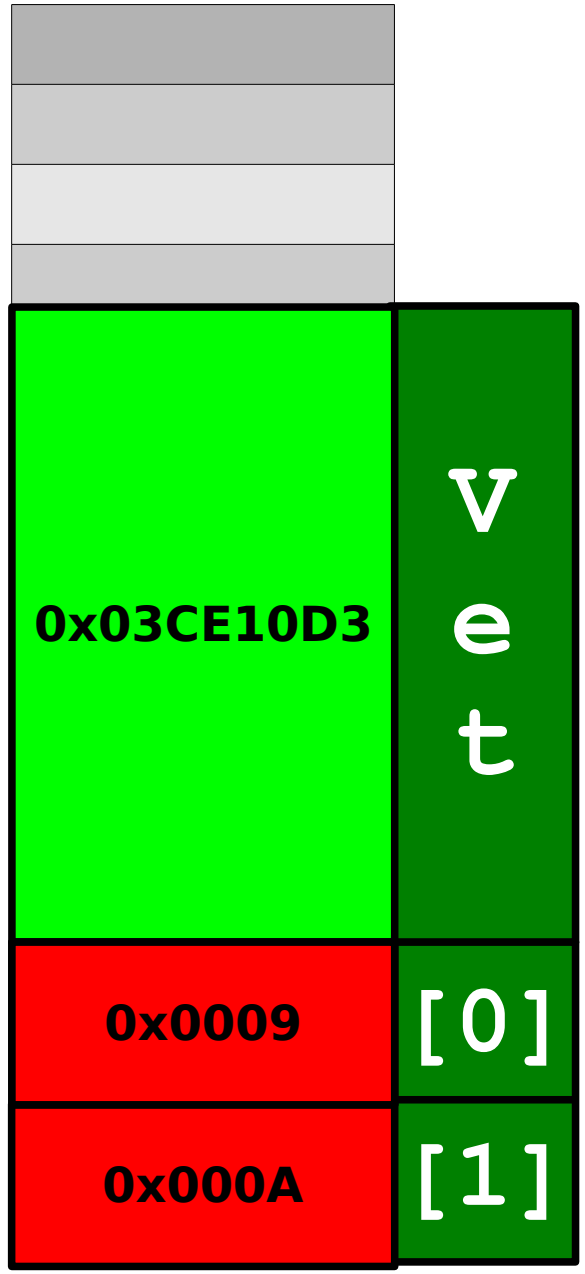
Alocação automática de memória

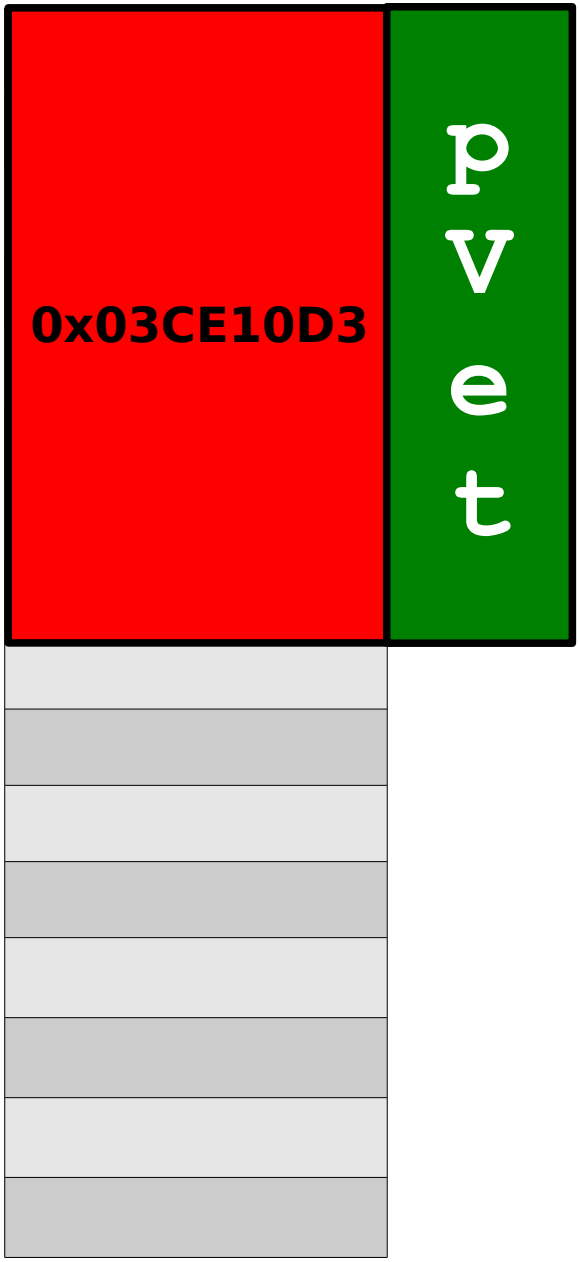
```
int main()
{
   char MyStr[5];
}
```



Apontadores e Arrays

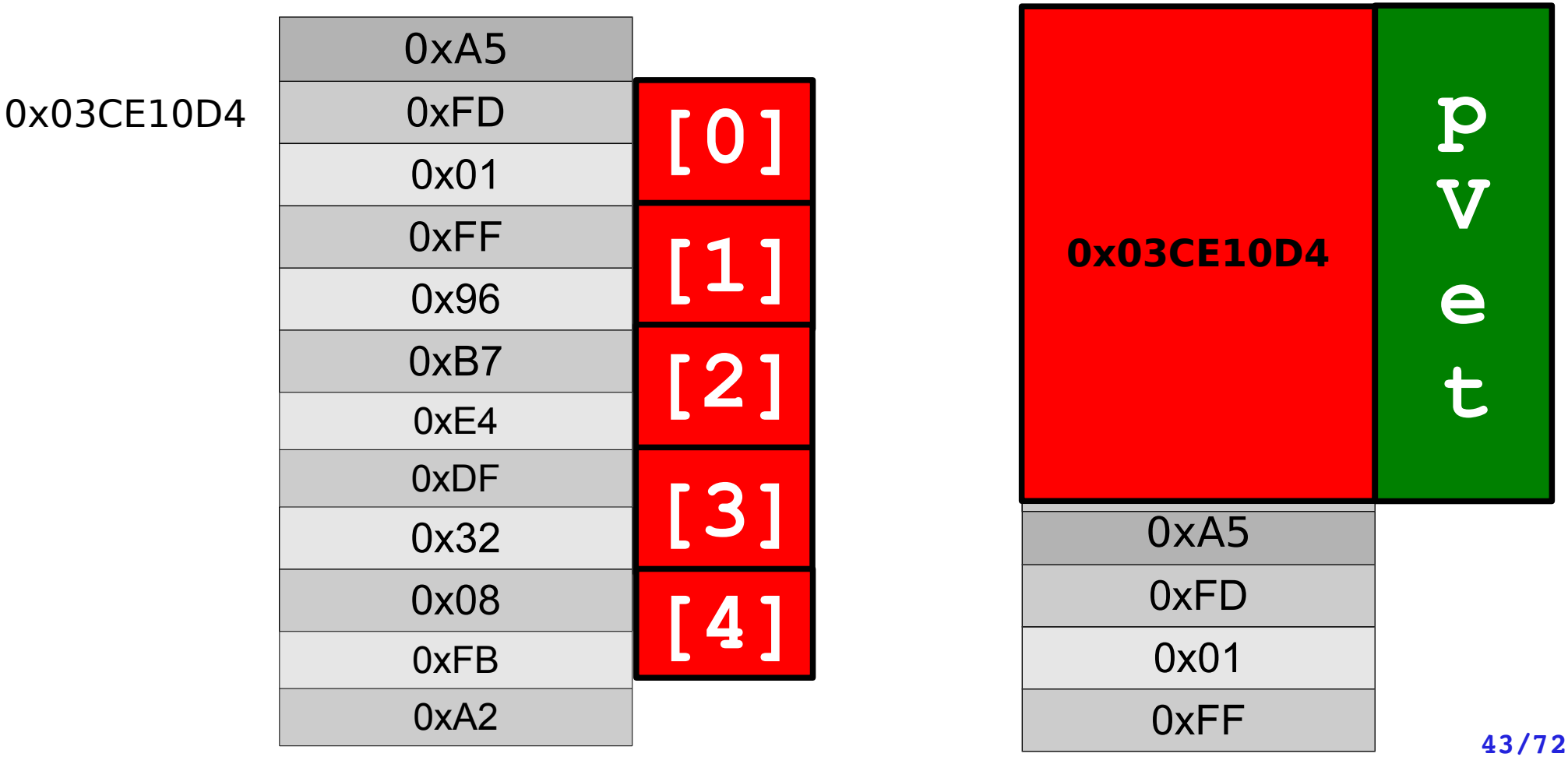
```
int main()
{
   short Vet[2];
   short* pVet;
   pVet = Vet;
   *pVet=5;
   pVet[0]=9;
   pVet[1]=10;
}
```





Alocação Dinâmica de memória

```
int main()
{
   short* pVet;
   pVet = (short*) malloc(5*sizeof(short));
}
```



Tratamento de caracteres em C

Funções da Biblioteca `ctype.h`

- Avaliação de caracteres:
  - `isalnum()` se é alfanumérico
  - `isalpha()` se é alfabético
  - `isblank()` se é branco
  - `iscntrl()` se is um caracter de controle
  - `isdigit()` se é um digito decimal
  - `isgraph()` se possui representação gráfica
  - `islower()` se é letra minúscula
  - `isprint()` se é imprimível
  - `ispunct()` se caracter é símbolo de pontuação

Funções da Biblioteca `ctype.h`

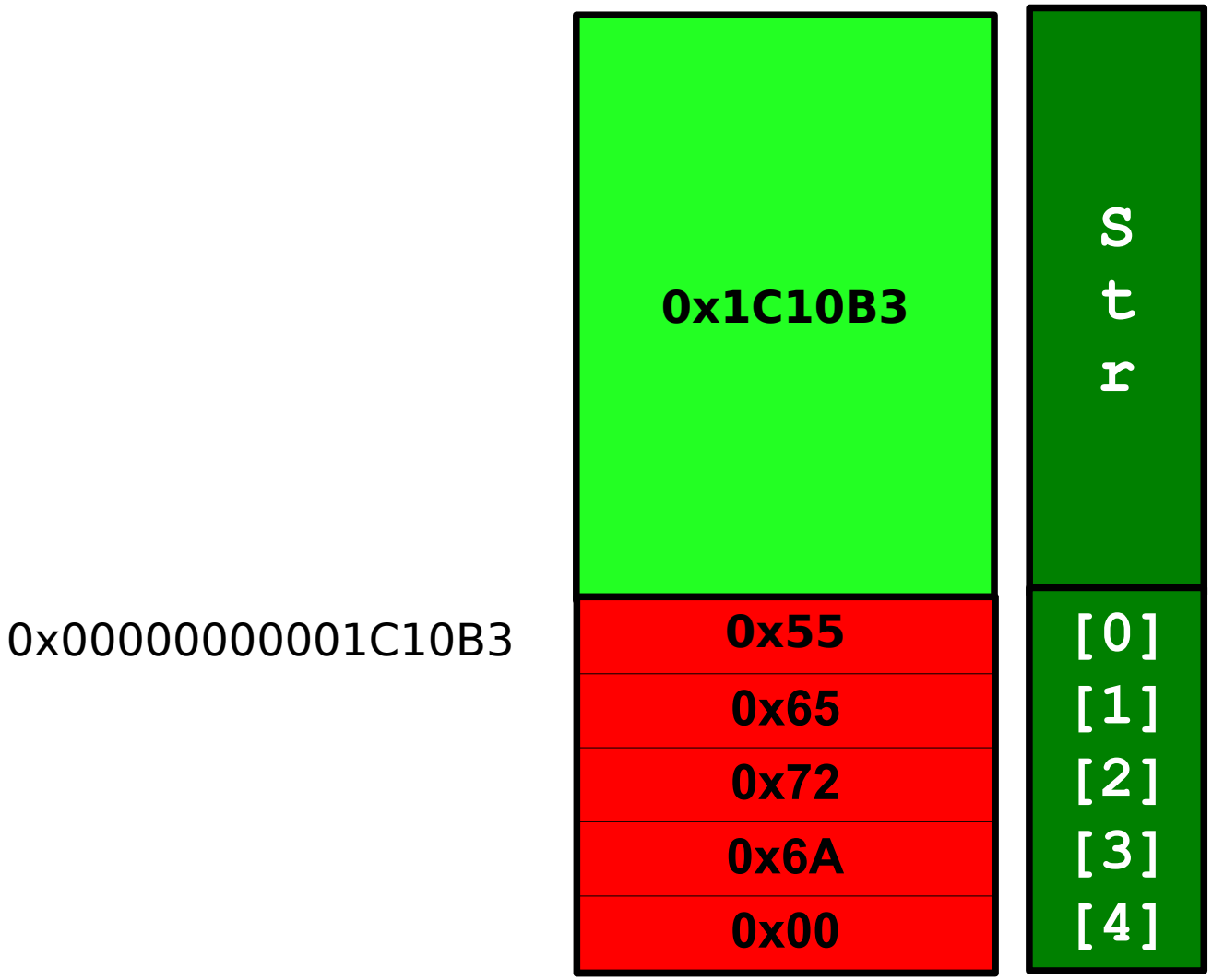
- Avaliação de caracteres:
  - `isspace()` se é maiúsculo espaço em branco
  - `isupper()` se é maiúsculo
  - `isxdigit()` se é um dígito hexadecimal válido

- Conversão de caracteres:
  - `tolower()` para minúsculo
  - `toupper()` para maiúsculo

String em C

String

```
char Str[5]="Uerj";
char Str[ ]="Uerj";
```



Funções da Biblioteca `string.h`

- Cópia:
  - `strcpy()` copia string
  - `strncpy()` copia caracteres de uma string
- Concatenation:
  - `strcat()` concatena strings
  - `strncat()` concatena parte de uma string a outra
- Comparison:
  - `strcmp()` compara duas strings caracter a caracter
  - `strcoll()` compara duas strings usando `LC_COLLATE`
  - `strncmp()` compara até *n* caracteres de duas strings
  - `strxfrm()` transforma string usando `LC_COLLATE`

Funções da Biblioteca `string.h`

- Busca:
  - `strchr()` a primeira ocorrência de um caracter
  - `strcspn()` fornece o comprimento da substring inicial de `str1` contendo somente caracteres que não pertençam a `str2`
  - `strpbrk()` idem `strcspn()`, retornando ponteiro
  - `strrchr()` a última ocorrência de um caracter
  - `strspn()` fornece o comprimento da substring inicial de `str1` contendo exclusivamente caracteres em `str2`
  - `strstr ()` localiza substring
  - `strtok ()` quebra string em tokens
- Outras:
  - `strerror()` obtém apontador para mensagem de erro
  - `strlen()` obtém o comprimento da string

Exercício U3.3 - Funções de Manipulação de Strings

Crie um programa que receba através do canal de entrada *default* uma string e teste se esta é um palíndromo.

`fgets (char * str,int num, FILE* stream)` lê uma string a partir do canal especificado

`strcmp (char * str1, char str2)` compara strings

Recursos complementares

Register

- Indica ao compilador que uma determinada variável será usada exaustivamente.
- Variáveis register não são armazenadas na memória principal, mas nos registradores localizados dentro do processador.
- Melhora o desempenho dos programas

```
register int x;

register char c;
```

const

- Especifica que o valor de uma variável ou parâmetro de uma função não poderá ser modificado.
- Num array `const` os valores dos elementos permanecem inalterados.

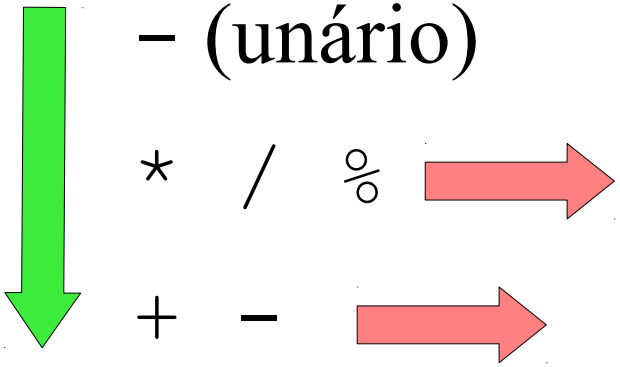
```
const double pi = 3.141592654;

const char c[]= "Guilherme é legal";
```

Operadores

Operadores Aritméticos

- Operadores binários
  - `+` soma
  - `-` subtração
  - `*` multiplicação
  - `/` divisão
  - `%` resto da divisão
- Operador unário
  - `-` negativo

- Precedência e ordem de avaliação



- Exemplo

```
2*25.2/32*-3
  = -4.725
```

Operadores Relacionais e Lógicos

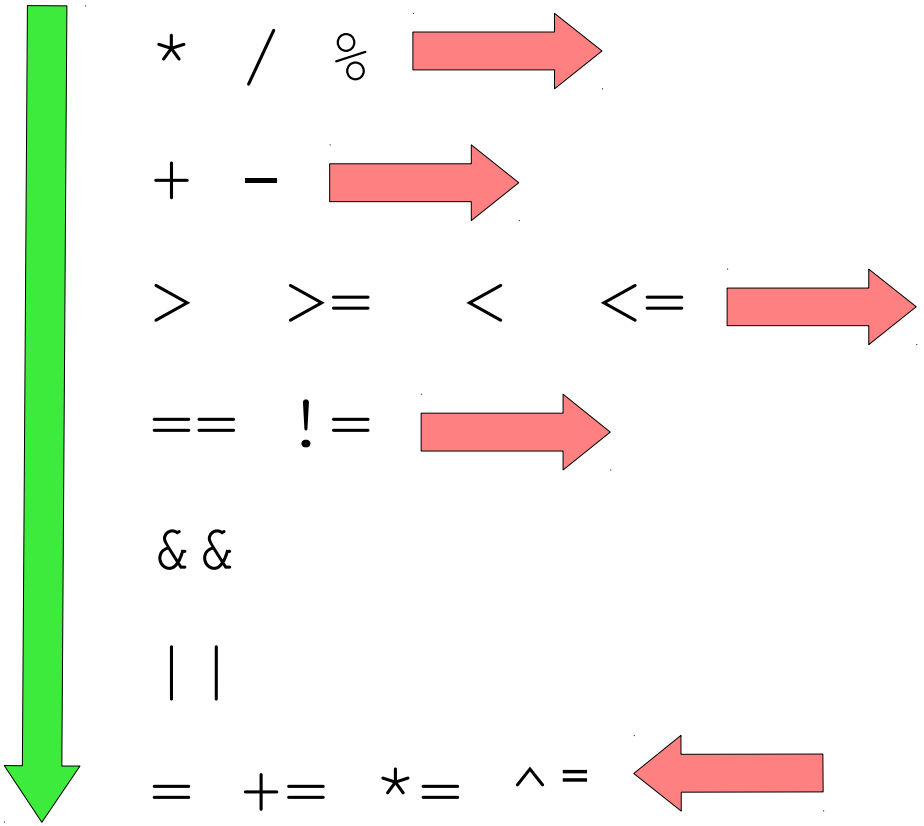
- Operadores binários

>	maior que
>=	maior ou igual
<	menor que
<=	menor ou igual
==	igual a
!=	diferente
&&	E lógico
||	OU lógico

- Operador unário

!	negação

- Precedência e ordem



Operadores Lógicos bit-a-bit

- Aplicáveis a operandos da família de inteiros:

```
char
short
int
long
long long
```

- Operadores binários
  - `&` E-Lógico
  - `|` OU-Lógico
  - `^` Ou-Exclusivo
- Operadores unários
  - `<<` *left shift*
  - `>>` *right shift*
  - `~` complemento

Operadores Lógicos bit-a-bit

- Exemplos

```
int n, x, y, z;

n = 0xFF & 0177; /* n = 0x7F */

x = 0xF00 | 0xFF; /* x = 0xFFF */

y = 0xFB << 2; /* y = 0x3EC */

z = ~ 0xFF; /* z = 0xFF00 */
```

Operadores de Atribuição

- Atribuição

```
n = 0xFF ;
```

- Incremento e decremento

```
n++; ++n; n--; --n;
```

- Atribuição + operador binário

```
n+=2; /* n=n+2
-=  *=  /=  %= <<= >>= &= ^= |= */
```

Operadores: Precedência e Ordem de Avaliação

Precedência dos operadores	Ordem de avaliação
`() [] -> .`	Esquerda → Direita
`! ~ ++ - + - * (type) sizeof`	Esquerda ← Direita
`* / %`	Esquerda → Direita
`+ -`	Esquerda → Direita
`<< >>`	Esquerda → Direita
`< <= > >=`	Esquerda → Direita
`== !=`	Esquerda → Direita
`&`	Esquerda → Direita
`^`	Esquerda → Direita
`|`	Esquerda → Direita
`&&`	Esquerda → Direita
`||`	Esquerda → Direita
`?:`	Esquerda ← Direita
`= += -= *= /= %= &= ^= |= <<= >>=`	Esquerda ← Direita
`,`	Esquerda → Direita

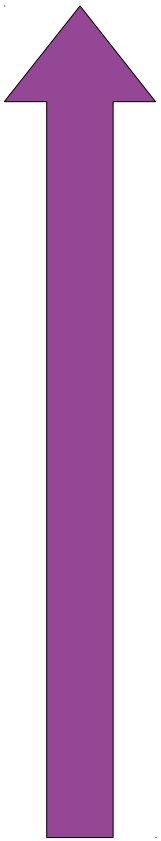
Conversões de Tipos

Conversões Automáticas

- Expressões aritméticas fazem conversões automáticas de tipos
  - `var1 <operador> var2`
  - se `var1` e `var2` são do `tipo1` o resultado é `tipo1`
  - se `var1` e `var2` é são de tipos diferentes, durante o cálculo um dos operandos é convertido
    - o resultado terá o mesmo para o qual o operando foi convertido

Operandos de Tipos Distintos

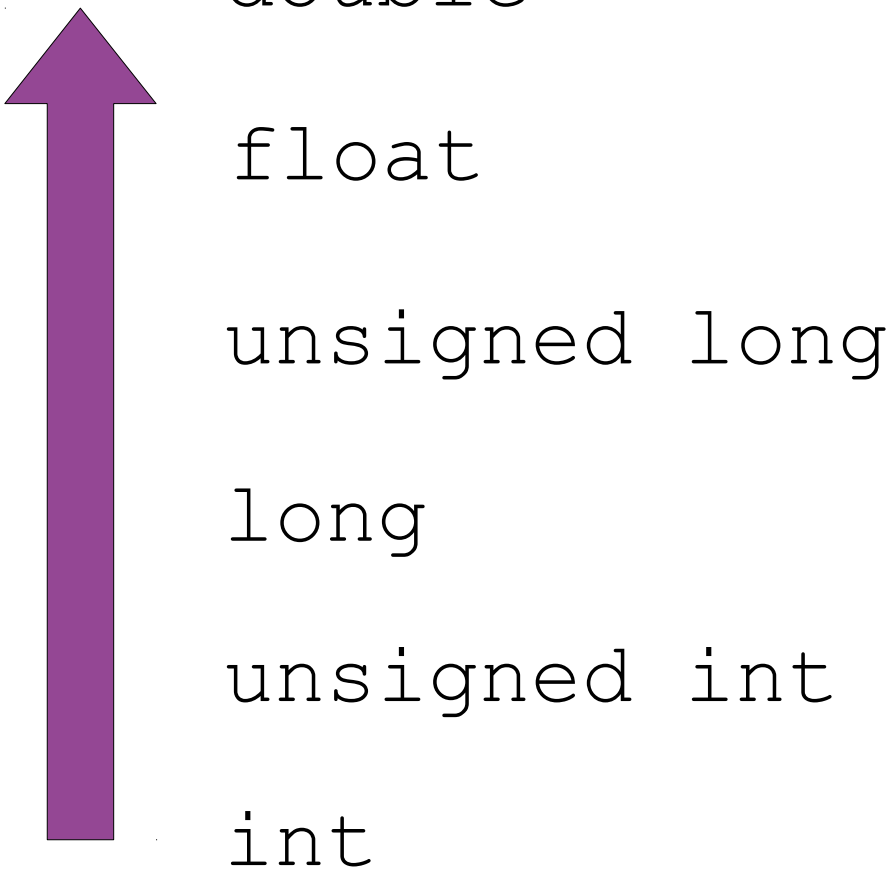
- Hierarquia de Conversão



```
double

float

unsigned long

long

unsigned int

int
```

Conversão Automática na Atribuição

- Do tipo menor para o tipo maior → OK



- Do tipo maior para o tipo menor → Truncamento

Conversão Automática do tipo `char`

- Todo `char` é convertido para `short`

```
/* atoi: convert s to integer */
int atoi(char s[])
{
   int i, n;
   n = 0;
   for (i = 0; s[i] >= '0' && s[i] <= '9'; ++i)
      n = 10 * n + (s[i] - '0');
   return n;
}
```

Conversão Explícita de Tipo

- O operador de conversão de tipo `(<tipo>)` muda o tipo do resultado da expressão a sua direita

```
(<tipo>) <expressão>
```

- O tipo da expressão se mantém
- O resultado é convertido para `<tipo>`

Compilação e Tabelas de Símbolos

Tabela de Símbolos

- Estrutura de dados, normalmente uma hash, que armazena informações sobre os identificadores
- Na compilação:
  - Monitoramento dos identificadores em uso
  - Análise léxica, sintática e semântica
- Arquivo objeto:
  - Mapeamento das variáveis em endereços relativos de memória
- *Linking*:
  - definição dos endereços de memória absoluta
- *Debuging* de Executáveis
- Conjunto de entradas em bibliotecas compartilhadas

Tabela de Símbolos

```
void myProc ( int A, short B)
{
   int D, E;
   D = 0;
   E = A / round(B);
   if ( E > 5)
      printf("%d", D);
}
```

Símbolo	Token	Dtype	Inicializado?	Endereço
myProc	id	procName	-	0x0E
A	id	int	SIM	0x00
B	id	short	SIM	0x04
D	id	int	SIM	0x06
E	id	int	SIM	0x0A

Trabalho U2.4 - Conversão de Base

Crie um programa que dada uma string contendo um número na base 8 (número octal) a converta para uma string relativa ao número na base 2 (binário) correspondente. Em seguida, seu programa deve converter a string com a representação binária em uma string com a representação hexadecimal.

Fim