Introdução à Programação de Computadores

Natan Sanches

Bacharelado em Ciências da Computação - ICMC/USP

Conteúdo Programado:

- Introdução e definição de um computador
- Arquitetura primitiva de Von Neumann
- A diferença entre memória Heap e Stack
- O que é um compilador?
- Linguagem compilada x Linguagem interpretada
- Introdução à manipulação de dados com C

Introdução e definição de um computador

O que torna computadores tão especiais? Ou melhor, o que classifica um dispositivo eletrônico como computador?

O que é um computador?

- Um computador é um conjunto de elementos eletrônicos (máquina) capaz de executar variados tipos de algoritmos e tratamento de informações (processamento de dados).
- Uma máquina que pode possuir inúmeros atributos, dentre eles o armazenamento de dados, processamento de dados, cálculo em grande escala, desenho industrial, etc.
- Trata-se de uma definição informal, pois a palavra “computador” possui um conceito matemático rigoroso por trás, utilizado em Teoria da Computação.
- Ícones da *Era da Informação*.

A máquina de Turing

- Caracterizou-se na união da lógica e da matemática em uma máquina.
- Trata-se de um sistema com uma cabeça de escrita/leitura que pode identificar símbolos em uma fita, potencialmente infinita, dividida em quadrados, um por vez.
- Para cada símbolo lido, o sistema é capaz de executar uma ação com base em uma lógica externa. Pode-se dizer que a máquina possui diferentes “estados”.

Vide: Teoria da Computação, Máquinas de Estado, Teoria dos Autômatos.

Representação minimalista de uma máquina de Turing, abstraída sua complexidade real.

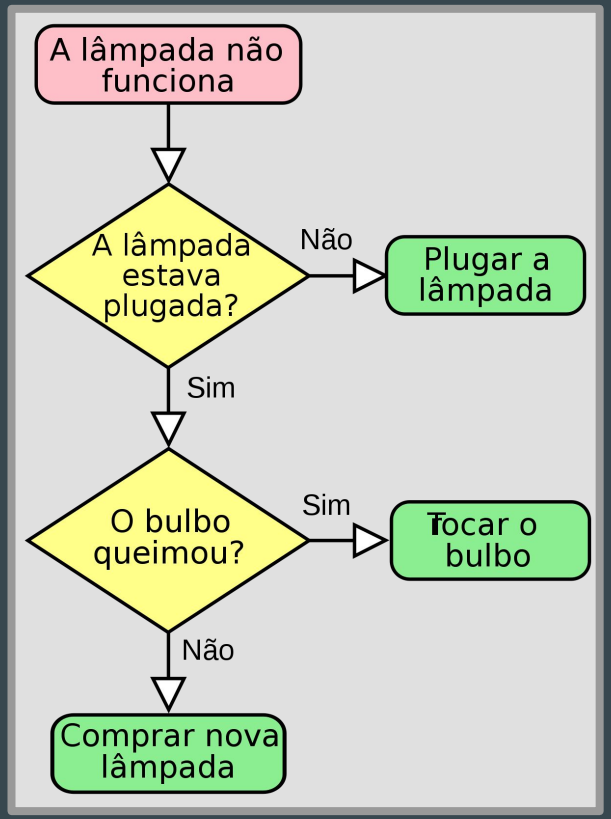
A máquina de Turing

O que é um um algoritmo?

- Na ciência da computação, um algoritmo pode ser descrito como uma série de processos discretos e sequenciais, que visam solucionar um dado problema dado de início.
- Um programa de computador é, em sua essência, um algoritmo que diz ao computador que passos realizar em dada situação.

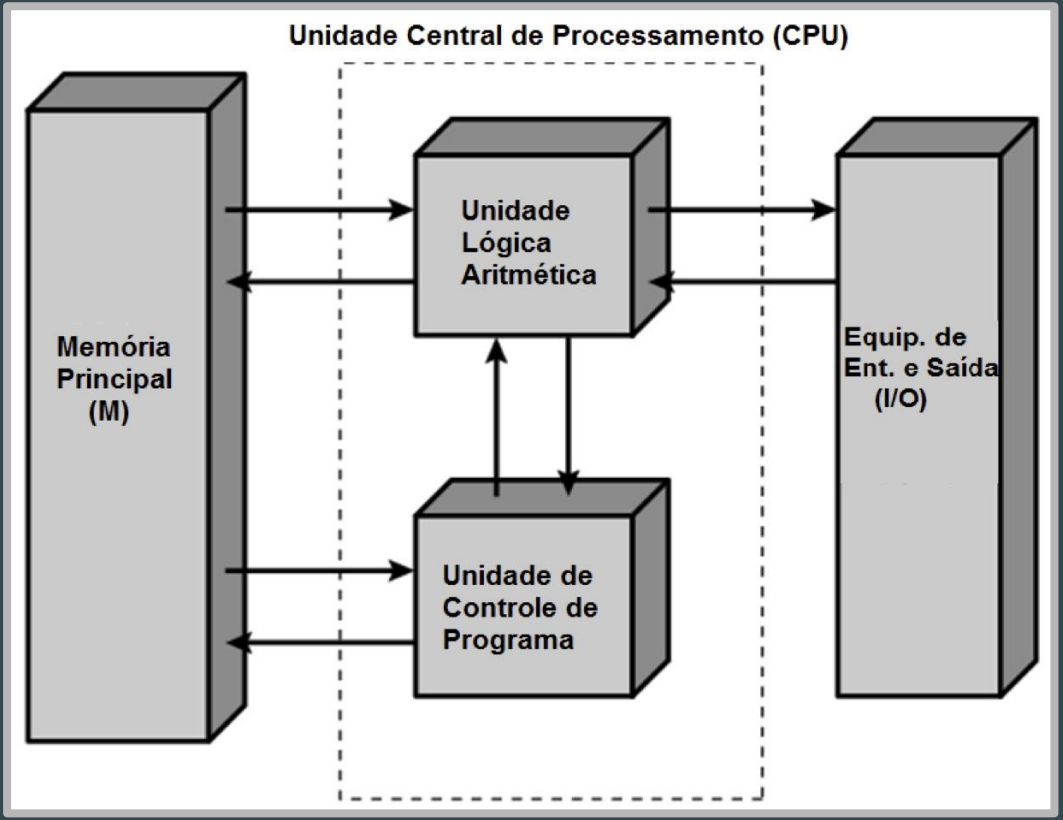
O que é um um algoritmo?

No fluxograma ao lado, a troca de lâmpadas descrita pode ser vista como um algoritmo. Ele também poderia, analogamente, ser escrito como um programa de computador.



A arquitetura de Von Neumann

Desde a elaboração teórica, até a construção de computadores modernos



Arquitetura de Von Neumann

Proposição de que os computadores fossem organizados em diversos componentes, onde cada um executa uma ação diferente.

Importâncias da Arquitetura de Von Neumann

- Introdução da utilização da *base binária* para o trabalho interno do hardware com números.
- Ancestral das arquiteturas modernas, que marcou o surgimento do que conhecemos hoje como computadores.
- Todos os componentes que formam o funcionamento do computador se tornam alinhados com a CPU, evitam-se problemas como dessincronização.

Dúvidas?

A segmentação da Memória RAM

A memória RAM não é homogênea: isto é, possui segmentos cujo sistema operacional delimita para seu funcionamento harmônico com o restante dos processos.

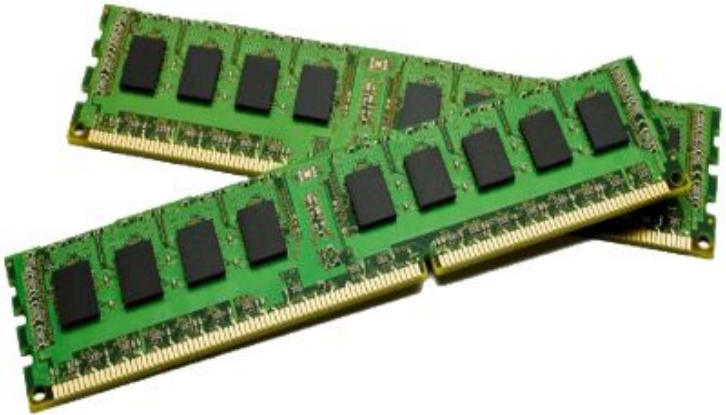
O que é memória?

O que é memória?

- Memória é um dos componentes internos do computador que permitem o armazenamento de dados (em forma de bits).
- Suas principais características são: volatilidade, velocidade de acesso, capacidade de armazenamento e possibilidade de leitura, escrita ou ambos.
- A velocidade de acesso costuma ser uma grandeza inversamente proporcional à capacidade de armazenamento, uma vez que em memórias com menor capacidade é possível um acesso mais rápido dos dados internos que a compõem.

Memória RAM

- Memória volátil
- Pouco armazenamento
- Permite leitura/escrita
- Acesso rápido e aleatório



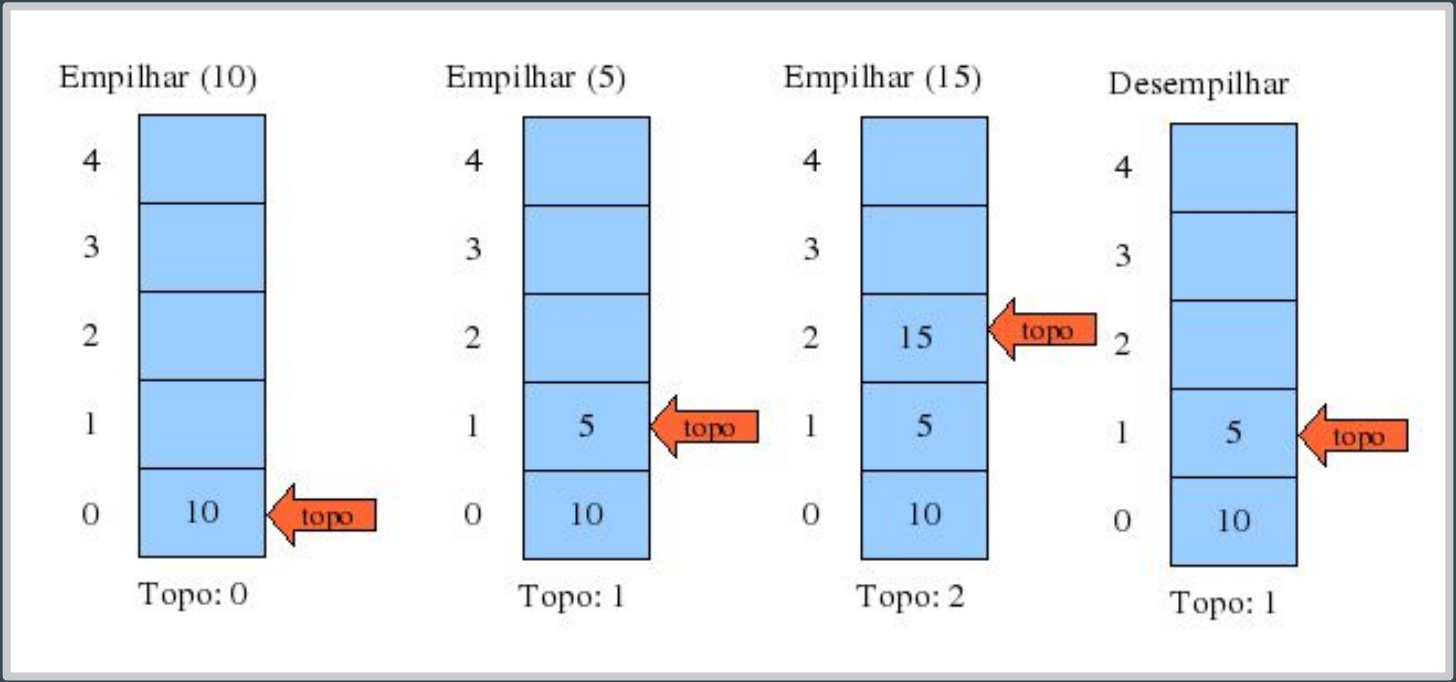
Segmentação da Memória RAM

Memória *Stack*

- Partição ‘limitada’ da memória RAM (normalmente, em torno de 8 Megabytes).
- Alocada pelo sistema operacional ao inicializar o programa.
- Armazena em formato de pilha.

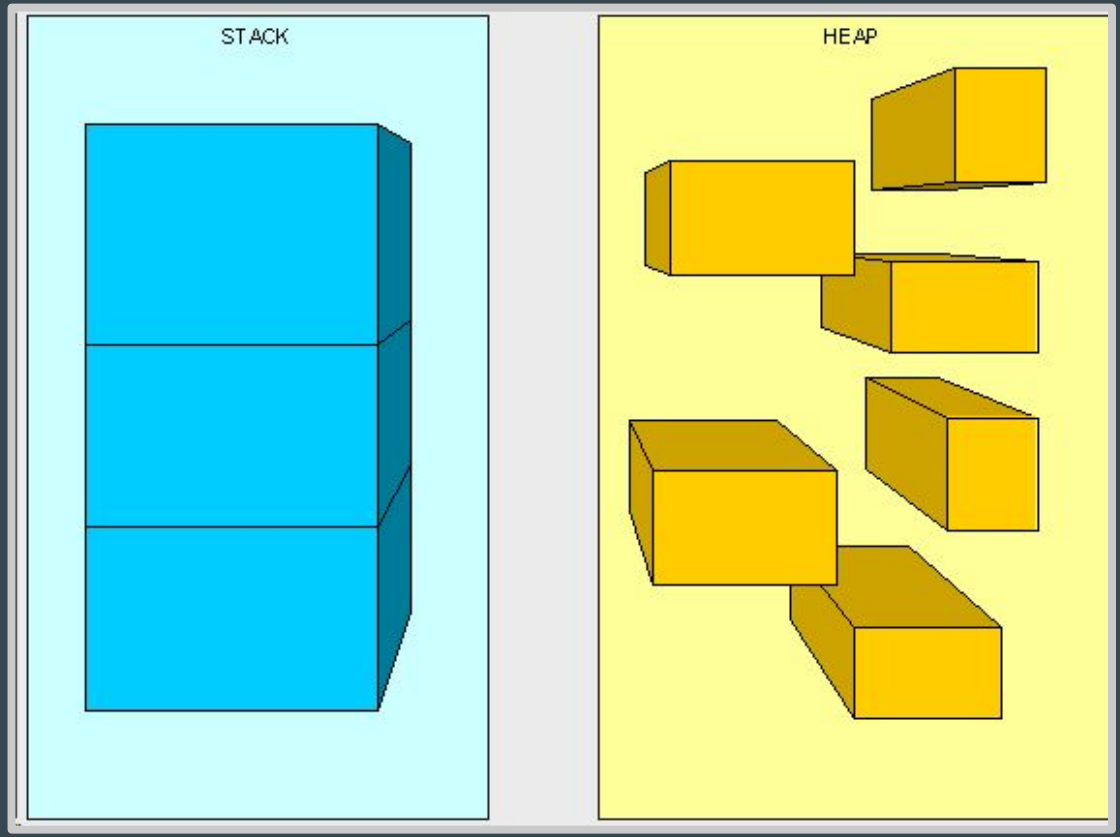
Memória *Heap*

- Partição majoritária da memória RAM, externa à memória Stack.
- Alocada e administrada pelo desenvolvedor no desenvolvimento do código.
- Armazenamento livre.

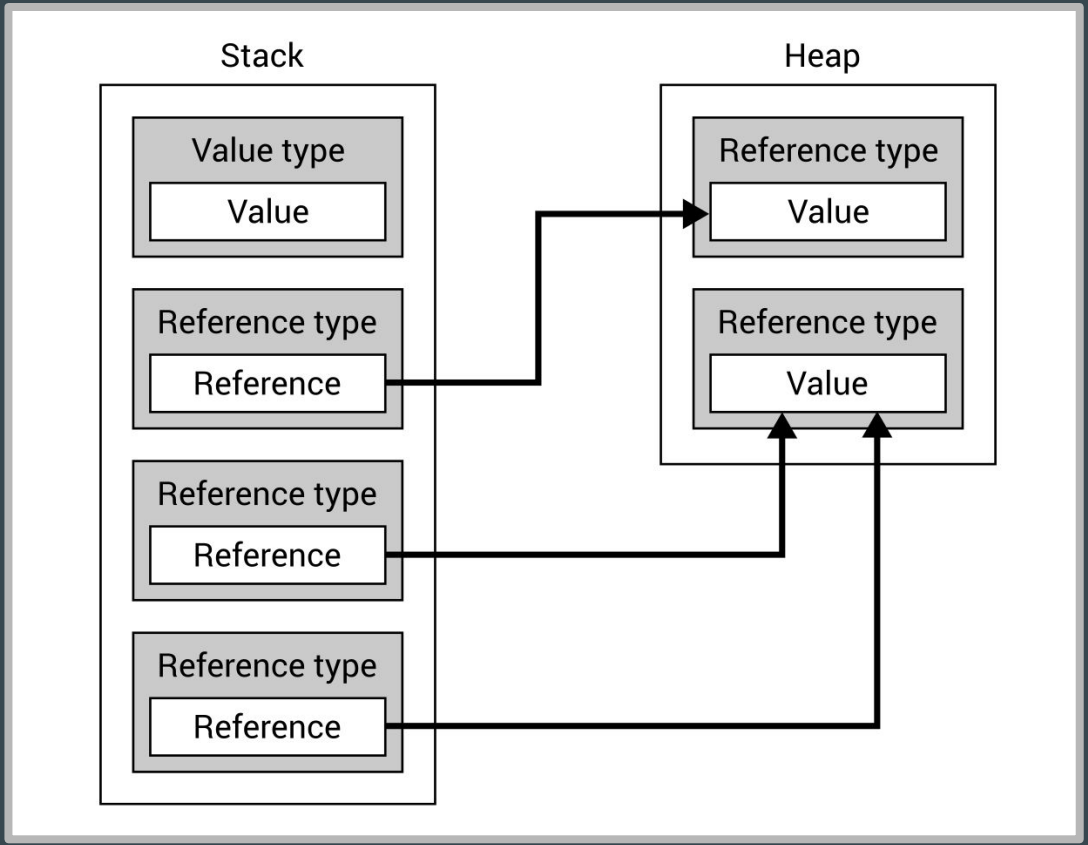


Representação do sistema LIFO (Last-In First-Out) em uma pilha

Exemplificação do processo de organização de ambos os segmentos.



Diferenciação entre os segmentos de memória



Em C, a relação da memória Heap com a Stack é ponteada pela utilização de variáveis ponteiros (ou *pointers*).

Integração das memórias *Heap* e *Stack*

A interpretação do computador à programas

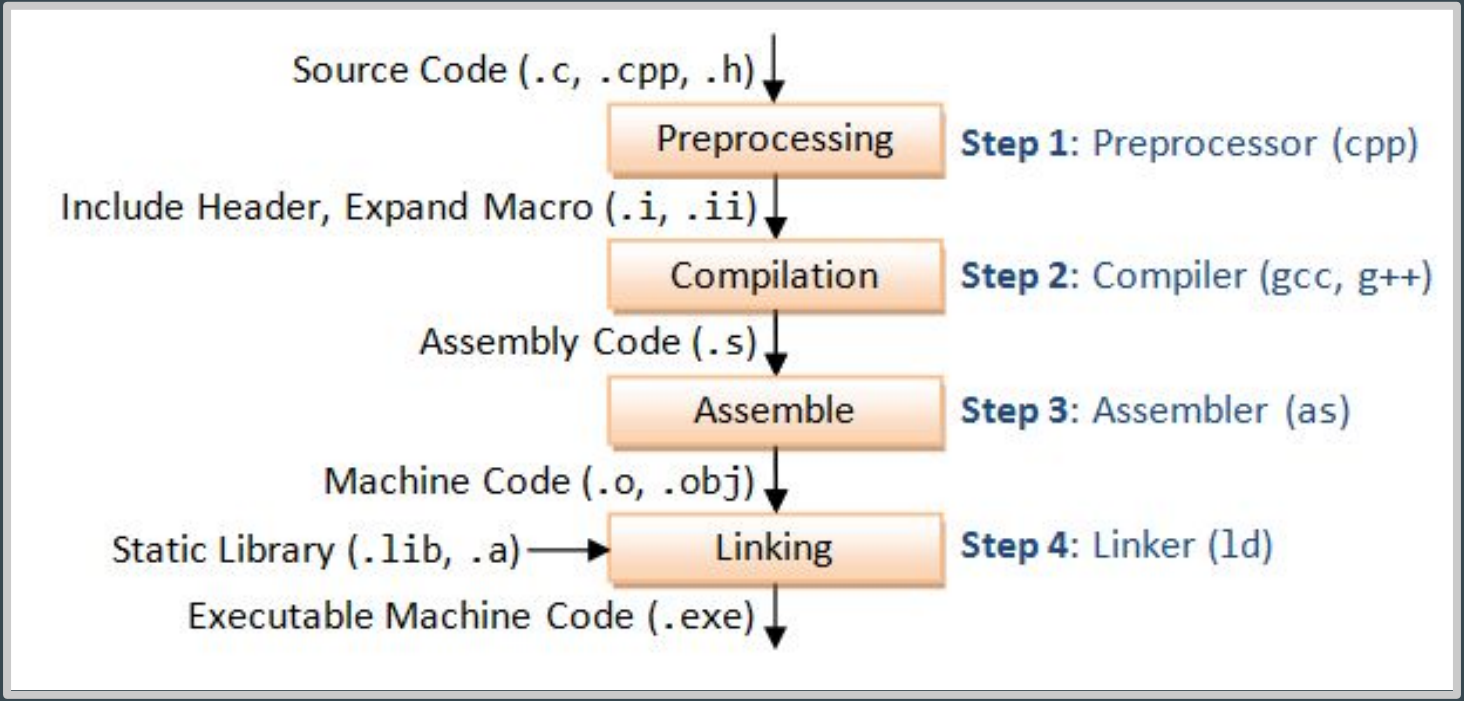
Como um computador reconhece as instruções de um código? É possível que um computador leia símbolos alfanuméricos?

NÃO!

Não é possível para um computador ler símbolos alfanuméricos diretamente.

O intermediário entre o hardware e o usuário

- O intermediário que permite a comunicação entre o hardware e o usuário se denomina **compilador**.
- Presente em todas as linguagens de programação que são *compiladas*.
- A função de um compilador é traduzir, a partir de uma linguagem de alto nível, o código-fonte de um arquivo para linguagem de máquina, que pode ser interpretada por uma CPU.
- Os compiladores também são escritos em linguagens de alto nível.



Processo de compilação de um código em C

Linguagem compilada x Linguagem interpretada

Linguagem compilada

- O compilador deve conhecer o tipo da variável antes da compilação.
- Todo o texto do código é traduzido para linguagem de máquina.
- Gera arquivos executáveis.

Linguagem interpretada

- Possui a vantagem das variáveis não-tipadas.
- O código-fonte é passado por um interpretador e executado no próprio.
- Não necessita de compilação.

Dúvidas?

Introdução à Programação em C

Por que aprender C?

- Deu origem a grande parte das linguagens de alto nível que conhecemos hoje (e.g.: Python). Seu aprendizado pode clarear o que acontece ‘por baixo dos panos’ nessas linguagens.
- Muito aplicada na construção de ferramentas de baixo nível, como drivers, compiladores, algoritmos de compressão e bancos de dados.
- Maior interação direta com o *hardware* e noção de *memory management*.

Introdução: bases numéricas

O entendimento de bases numéricas é fundamental para o estudo da computação.

- A **base** de um sistema de numeração é uma certa quantidade de unidades que constituirá uma quantidade de ordem superior.

Exemplos:

- **Binária** (2): 0, 1.
- **Octal** (8): 0, 1, 2, 3, 4, 5, 6, 7.
- **Decimal** (10): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.
- **Hexadecimal** (16): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Aritmética de bases

No cotidiano, estamos acostumados a utilizar a base decimal para efetuar cálculos e representar expressões matemáticas. No entanto, a maneira com que realizamos as operações aritméticas fundamentais (soma e multiplicação) são idênticas para quaisquer bases.

- **Decimal:** 5 + 6 = 11. (leia-se: *onze*)
- **Binária:** 1 + 1 = 10. (leia-se: *um zero*)
- **Hexadecimal:** E + 4 = 12 (leia-se: *um dois*)

Dúvidas?

Tipos de dados da linguagem C

Ao contrário de uma linguagem interpretada, a linguagem C é compilada. Isso indica que devem ser atribuidos tipos às variáveis declaradas.

Tipos de dados da linguagem C

- Os tipos de dados se manifestam na declaração de uma variável.
- Refere-se por tipo de dado: indicar ao compilador o que está sendo armazenado internamente àquela variável, de maneira com a qual ele possa reconhecer como tratá-la.
- Um mesmo tipo de dado pode receber diferentes tratamentos e produzir resultados diferentes, de acordo com qual modo a variável que o armazena fora declarada.

Tipos de dados primitivos

Palavra-chave	Tipo de variável	Interpretação
char	Caractere	Escrita de símbolos alfanuméricos.
int	Inteiro	Operações aritméticas no conjunto dos números inteiros.
float	Real de precisão simples	Operações aritméticas no conjunto dos números reais.
double	Real de precisão dupla	Operações aritméticas com números reais de muitas casas decimais.
void	Vazio (n/a)	Declaração de variáveis não-tipadas.

O tipo caractere

- É armazenado em apenas 1 byte (8 bits).
- Cada símbolo possui um valor representado em uma tabela de encoding (ASCII, Unicode, ...)
- Um tipo caractere também pode ser interpretado como inteiro: Em C, o inteiro representativo será o valor representado pelo símbolo na tabela ASCII.

- Cada símbolo representado visualmente possui um representante inteiro.

	00	16	32	48	64	80	96	112
0	NUL	DLE		0	@	P	`	p
1	SOH	DC1	!	1	A	Q	a	q
2	STX	DC2	"	2	B	R	b	r
3	ETX	DC3	#	3	C	S	c	s
4	EOT	DC4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SYN	&	6	F	V	f	v
7	BEL	ETB	'	7	G	W	g	w
8	BS	CAN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
10	LF	SUB	*	:	J	Z	j	z
11	VT	ESC	+	;	K	[	k	{
12	FF	FS	,	<	L	\	l	|
13	CR	GS	-	=	M	]	m	}
14	SO	RS	.	>	N	^	n	~
15	SI	US	/	?	O	_	o	DEL

Tabela ASCII (simplificada)

Introdução à aritmética de números

- Internamente, todos os números são representados como binários e suas operações são feitas, também, em sua forma binária.
- A necessidade de representação de números como *bits* requer que determinemos uma quantidade de bits para armazenamento desses dados. Esse fato traz consigo possíveis problemas de precisão.
- e.g.: Um inteiro de 8 bits (1 byte) possui 2^8 possíveis combinações de zeros e uns. Dessa forma, esse inteiro suportará atribuição de um intervalo simétrico entre números negativos e não-negativos: [-128, 127].

O tipo inteiro

- No início, o tamanho de um inteiro era determinado pela arquitetura do computador.
- Um mesmo inteiro poderia ter um valor diferente quando compilado em máquinas diferentes.

Na linguagem C

- Um inteiro pode ser representado por uma sequência de 8, 16, 32 ou 64 bits.
- Na linguagem C, um *int* tem representação padronizada de 4 bytes (32 bits).

O tipo inteiro

Bits	Inteiro com sinal	Intervalo	Inteiro sem sinal	Intervalo
8	`int8_t`	-128 a 127	`uint8_t`	0 a 255
16	`int16_t`	-32.768 a 32.767	`uint16_t`	0 a 65.535
32	`int32_t`	(- 2^31) a (2^31 - 1)	`uint32_t`	0 a 4.294.967.295
64	`int64_t`	(- 2^63) a (2^63 - 1)	`uint64_t`	0 a (2^64 - 1)

Quanto de memória seria necessária para a representação completa do número π?

INFINITA!

O que é ponto flutuante?

- Ponto flutuante é o formato de representação de números decimais (i.e. reais) utilizado pelos computadores modernos.
- A precisão de um computador possui limites: para reais de dezenas de casas decimais, um computador terá dificuldade de processar todos os seus dígitos.
- Os tipos relacionados à ponto flutuante são os tipos *float* e *double*.

O que é ponto flutuante?

Float (binary32)**:** número real com precisão simples. Possui aproximadamente 6 ou 7 dígitos de precisão.

- Ocupa 4 bytes na memória.

Double (long float / binary64)**:** número real com precisão dupla. Para números de muitas casas decimais. Abrange até 16 casas decimais de precisão.

- Ocupa 8 bytes na memória.

Tipos complementares e prefixos

- **Long:** Aumenta o intervalo numérico de representação de um tipo primitivo de dado.
- **Short:** Reduz o intervalo numérico de representação de um tipo primitivo de dado.
- **Unsigned:** Converte a parte negativa do intervalo numérico de um tipo primitivo como uma extensão da abrangência de números positivos.