PROVA DIDÁTICA

Prof. Orlando Saraiva do Nascimento Júnior
orlando.saraiva@unesp.br
orlando.nascimento@fatec.sp.gov.br

"The only way to learn a new programming language is by writing programs in it."

Dennis Ritchie

Apresentação da aula

Objetivos

Conhecer a estrutura de dados Pilha

- Implementar pilha em C com uso de vetor
- Implementar pilha em C de forma dinâmica

Conhecer a estrutura de dados Fila

- Implementar fila em C com uso de vetor
- Implementar fila em C de forma dinâmica

Pilha

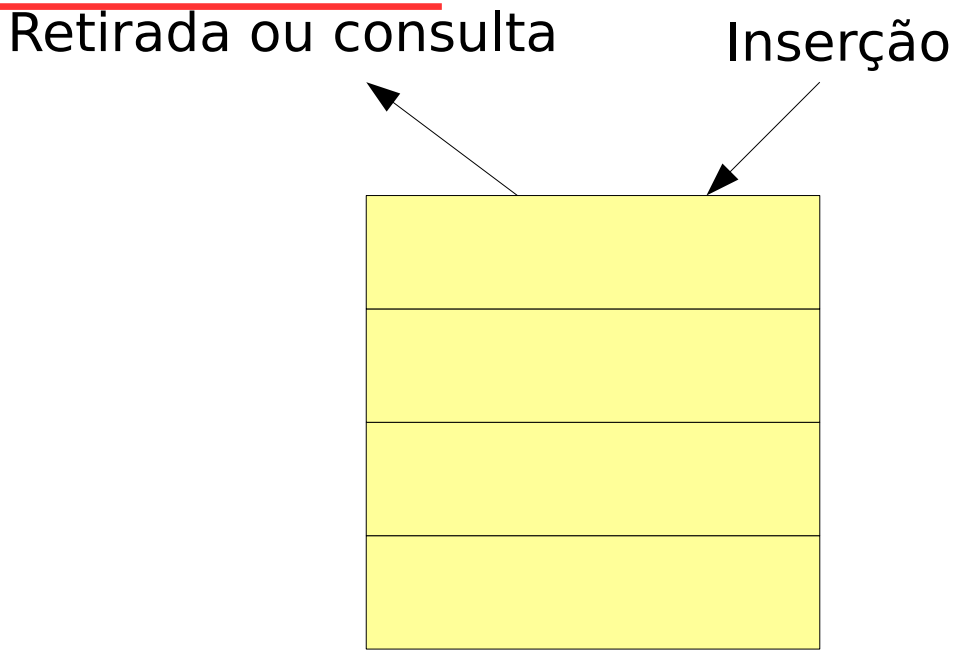
Pilha

Uma **pilha** é um conjunto ordenado de itens no qual novos itens podem ser inseridos a partir do qual podem ser eliminados itens em uma extremidade chamada **topo** da pilha.

Ao contrário do que acontece com o vetor, a definição de pilha compreende a inserção e a eliminação de itens, de moto que uma pilha é um objeto dinâmico, constantemente mutável.

Quando um item é incluído numa pilha, ele é **empilhado** sobre a pilha e, quando um item é removido, ele é **desempilhado.**

Pilha



O primeiro item a ser inserido na pilha é o último a ser removido.

Esta política é conhecida pela sigla LIFO (**L**ast **I**n **F**irst **O**ut)

Sempre que houver uma remoção, o elemento removido é o que está na estrutura há menos tempo.

Exemplos

Exemplo 1

Navegadores de Internet armazenam os endereços dos sites visitados recentemente em uma pilha. Cada vez que um usuário visita um novo site, o endereço é “enviado” para a pilha de endereços. O navegador permite o usuário desempilhe o último endereço acessado via botão voltar.

Exemplo 2

Os editores de texto geralmente fornecem um mecanismo de “desfazer” que cancela operações de edição e reverte para estados anteriores de um documento. Esta operação de desfazer pode ser obtida mantendo as alterações de texto em uma pilha.

Pilhas

As pilhas são as estruturas de dados mais simples, mas também estão entre as mais importantes.

Formalmente, uma pilha é um tipo de dado abstrato (ADT) tal que uma instância S suporta as seguintes funcionalidades:

S.push(elem) → Adiciona o elemento elem ao topo da pilha S

S.pop() → Remove e retorna do topo da pilha S.

Espera-se um erro caso a pilha esteja vazia.

Pilhas

No exemplo, teremos outros três funcionalidades

S.top() → Retorna a referência do topo da lista, sem remover o elemento da lista.

S.is_empty() → Returna verdadeiro, caso a pilha esteja vazia.

S.tamanho() → Returna o número de elementos da pilha S.

Pilhas

Operação	Valor de retorno	Conteúdo da pilha
S.push(5)	-	[5]
S.push(3)	-	[5, 3]
S.tamanho()	2	[5, 3]
S.pop()	3	[5]
S.is_empty()	False	[5]
S.pop()	5	[ ]
S.is_empty()	True	[ ]
S.pop()	“erro”	[ ]
S.push(7)	-	[7]
S.push(9)	-	[7, 9]
S.push(4)	-	[7, 9 , 4]
S.tamanho()	3	[7, 9 , 4]
S.push(8)	-	[7, 9 , 4 , 8]

Pilhas

Operação	Valor de retorno	Conteúdo da pilha
S.push(6)	-	[7 , 9 , 4 , 8 , 6]
S.push(1)	-	? ?
S.tamanho()	6	? ?
S.pop()	1	? ?
S.is_empty()	False	? ?
S.pop()	6	? ?
S.pop()	8	? ?
S.pop()	4	? ?
S.pop()	9	? ?
S.push(1)	-	? ?
S.pop()	1	? ?
S.pop()	7	[ ]
S.pop()	“erro”	[ ]

Pilhas

Operação	Valor de retorno	Conteúdo da pilha
S.push(6)	-	[7 , 9 , 4 , 8 , 6]
S.push(1)	-	[7 , 9 , 4 , 8 , 6 , 1]
S.tamanho()	6	[7 , 9 , 4 , 8 , 6 , 1]
S.pop()	1	[7 , 9 , 4 , 8 , 6]
S.is_empty()	False	[7 , 9 , 4 , 8 , 6]
S.pop()	6	[7 , 9 , 4 , 8]
S.pop()	8	[7 , 9 , 4]
S.pop()	4	[7 , 9]
S.pop()	9	[7]
S.push(1)	-	[7, 1]
S.pop()	1	[7]
S.pop()	7	[ ]
S.pop()	“erro”	[ ]

Implementando pilha com C
Primeira versão

Prof. Me. Orlando Saraiva Júnior

Primeira versão

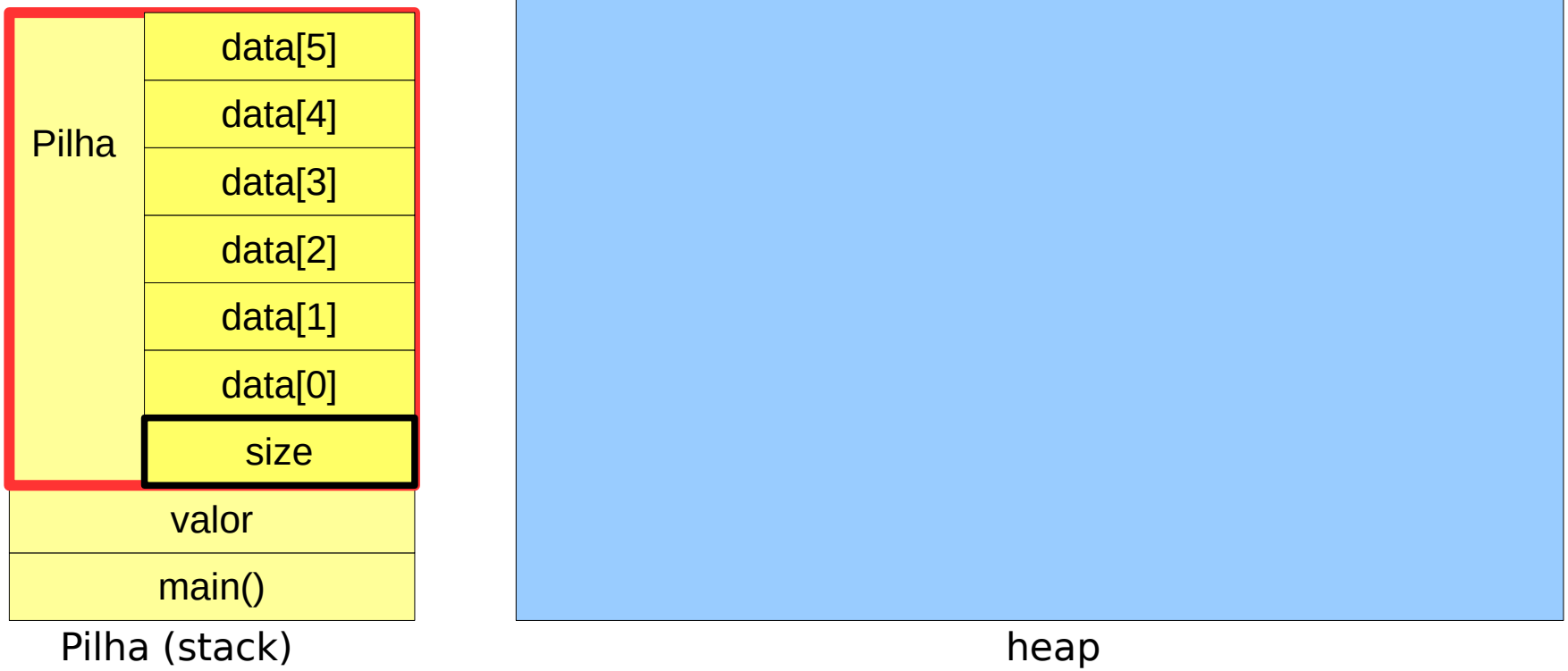
Há várias formas de se representar uma pilha. Examinaremos a mais simples delas.

Uma pilha é um conjunto ordenado de itens, e em C já contém um tipo de dado que representa um conjunto ordenado de itens: o vetor.

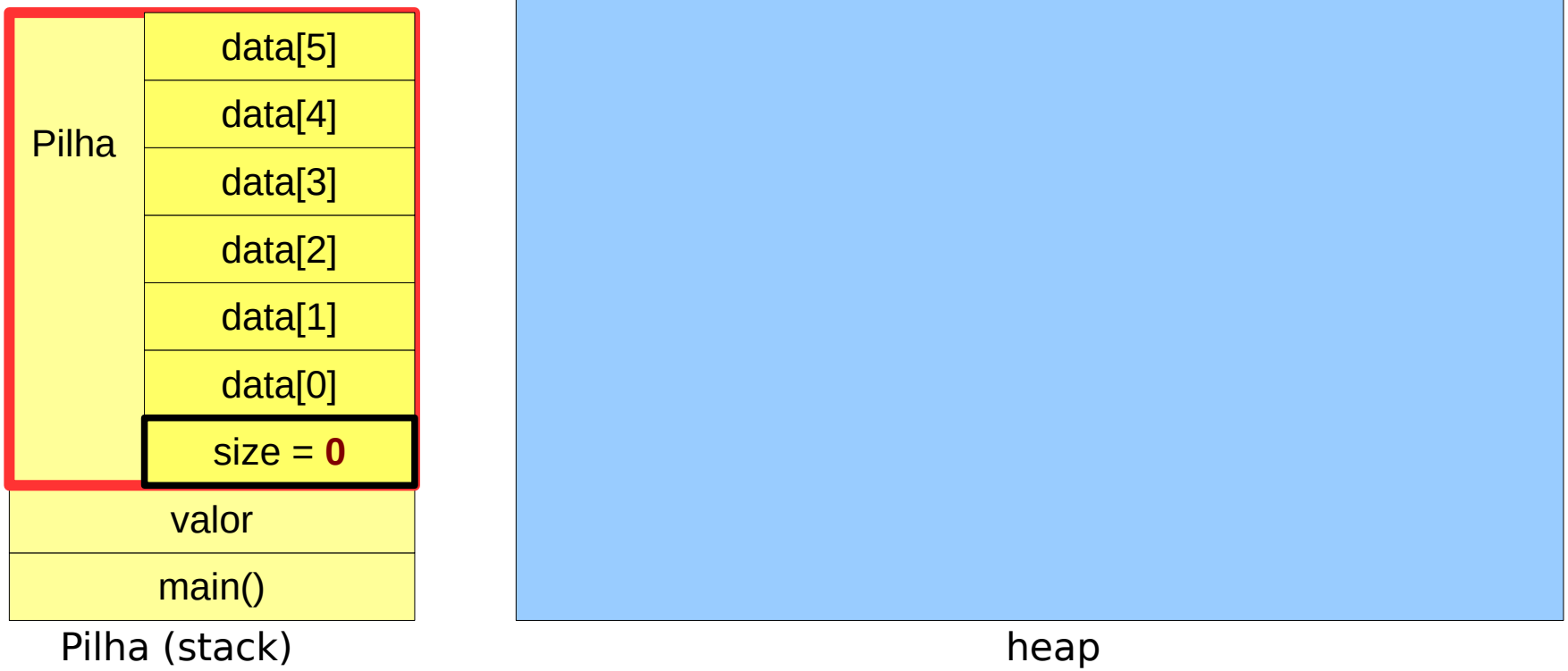
Contudo, um vetor e uma pilha são entidades totalmente diferentes. O número de elementos de um vetor é fixo. Em termos gerais, o usuário não pode alterar este número. Uma pilha é um objeto dinâmico, cujo tamanho está sempre mudando, a medida que os itens são empilhados e desempilhados.

Uma pilha em C pode ser declarada como uma estrutura contendo dois objetos: um vetor para armazenar os elementos e um inteiro para indicar a posição da pilha.

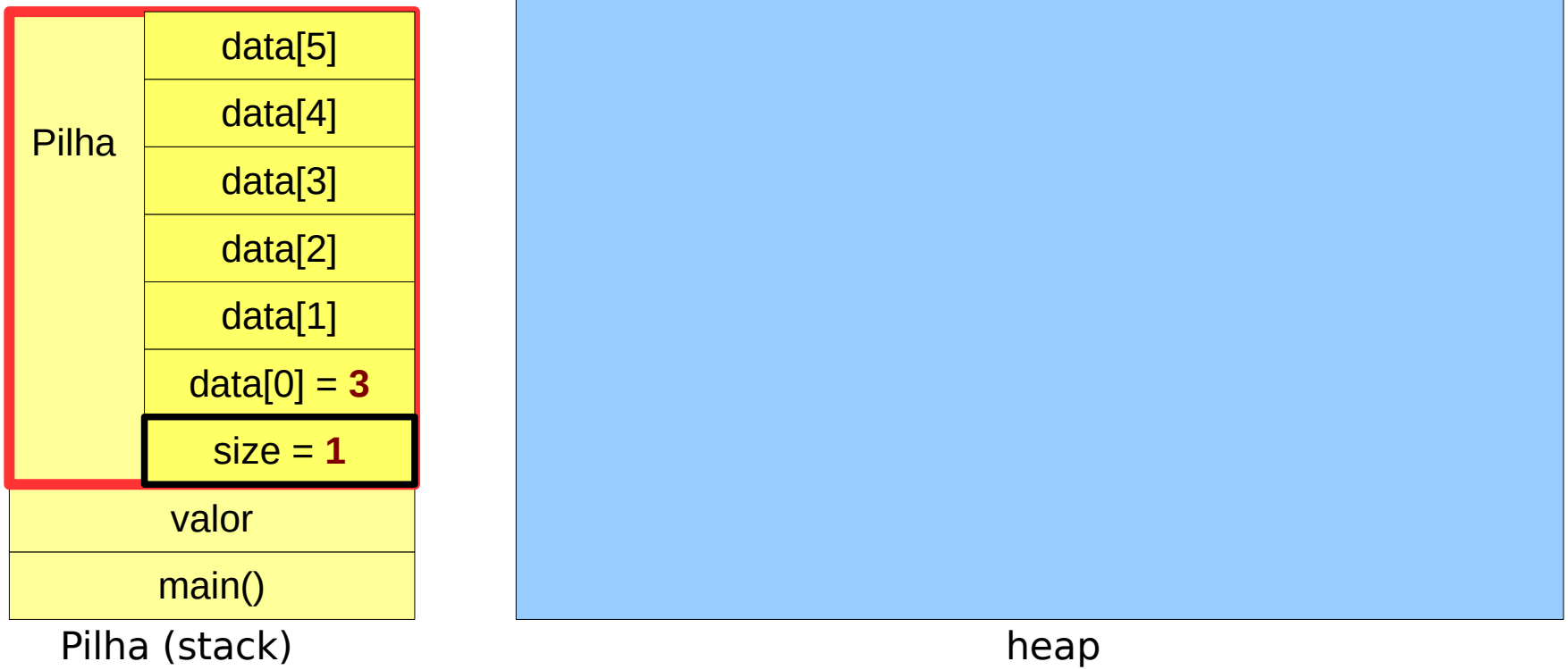
pilha1.c
Variáveis



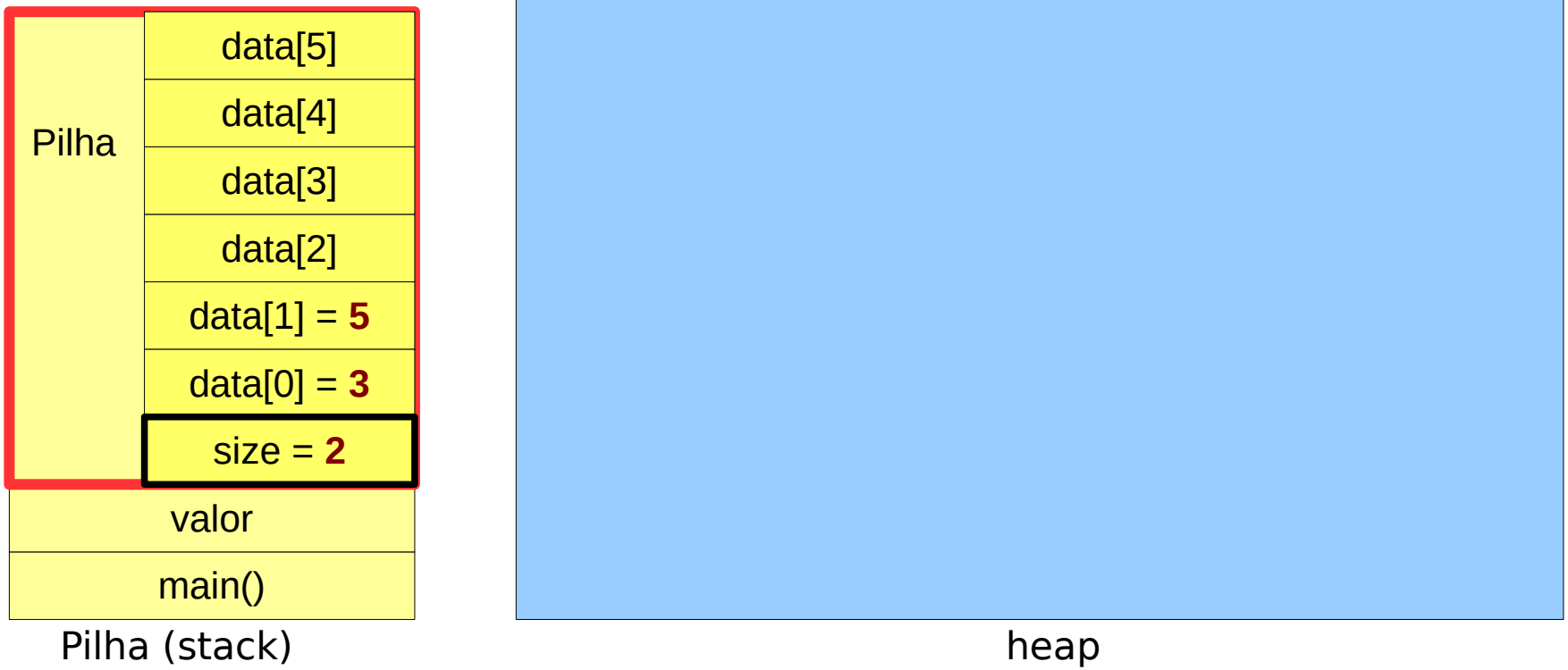
pilha1.c
Variáveis



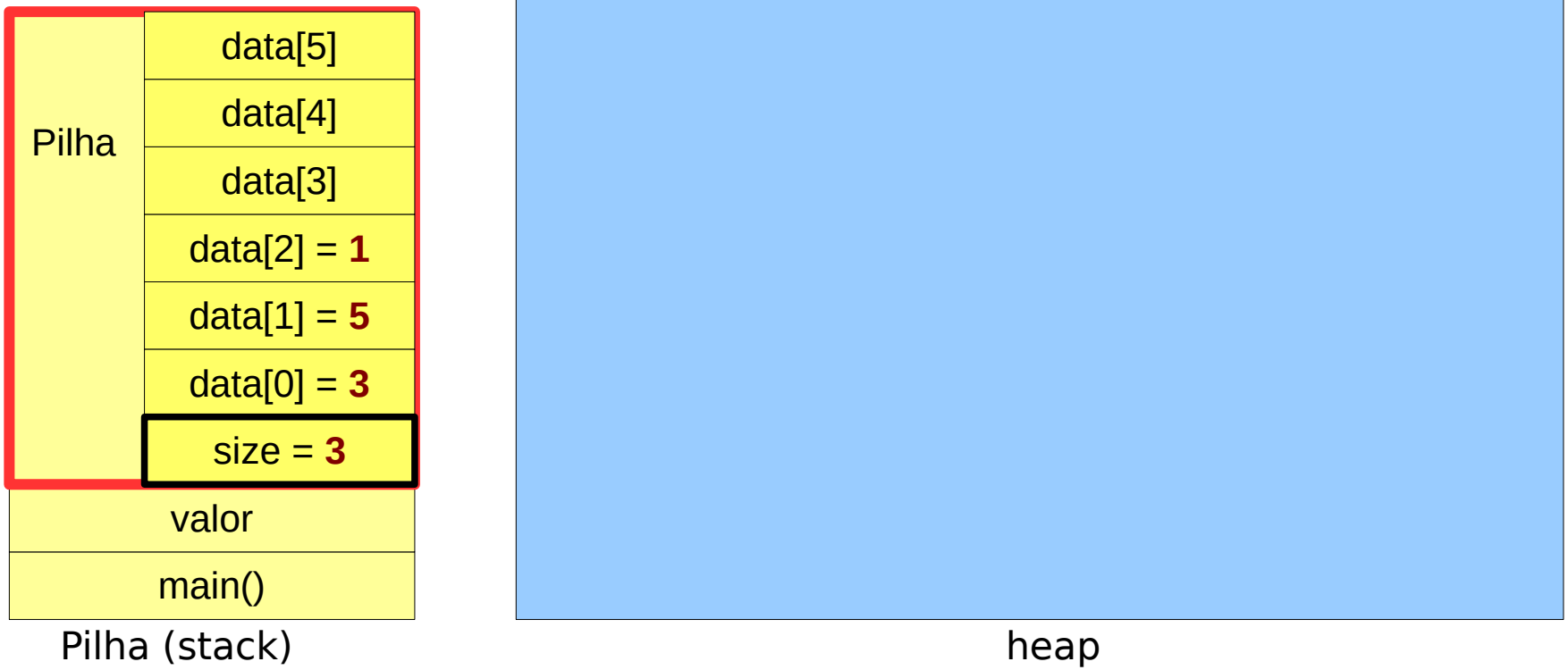
pilha1.c
Variáveis



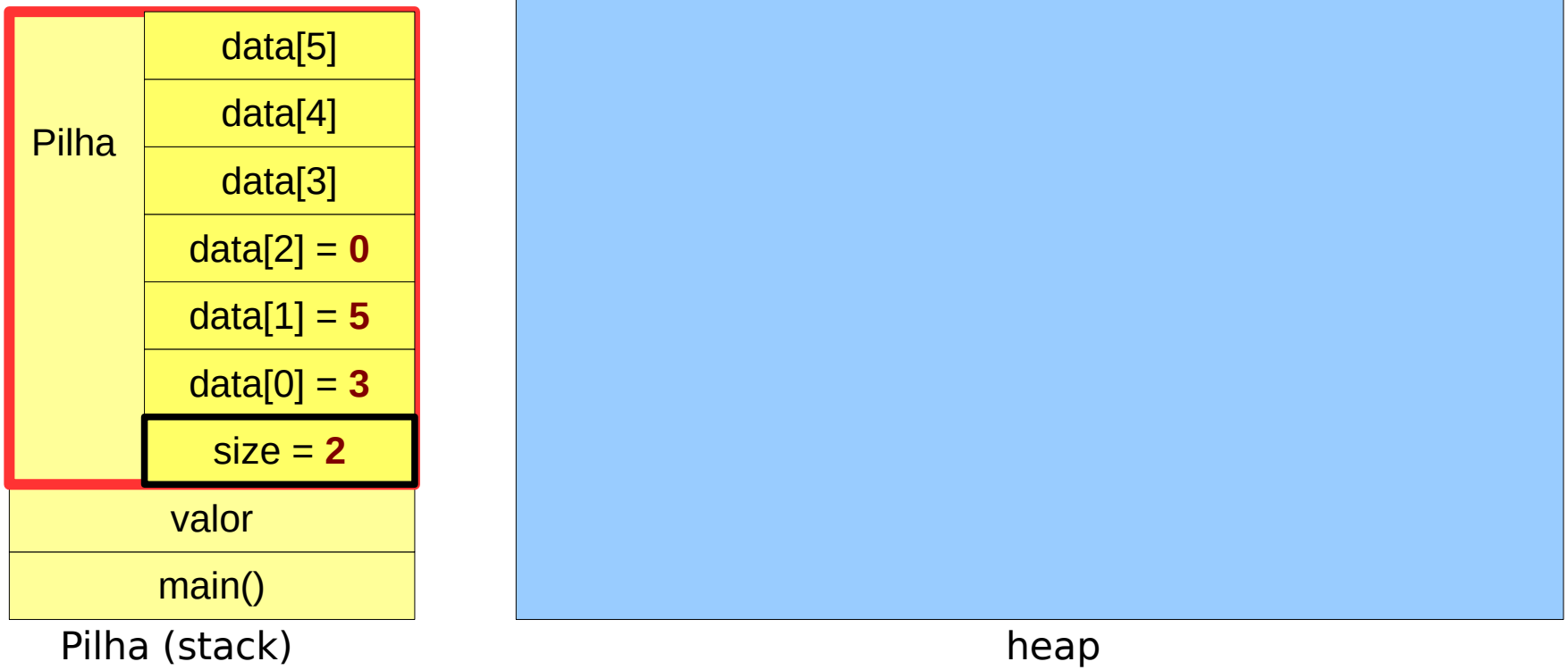
pilha1.c
Variáveis



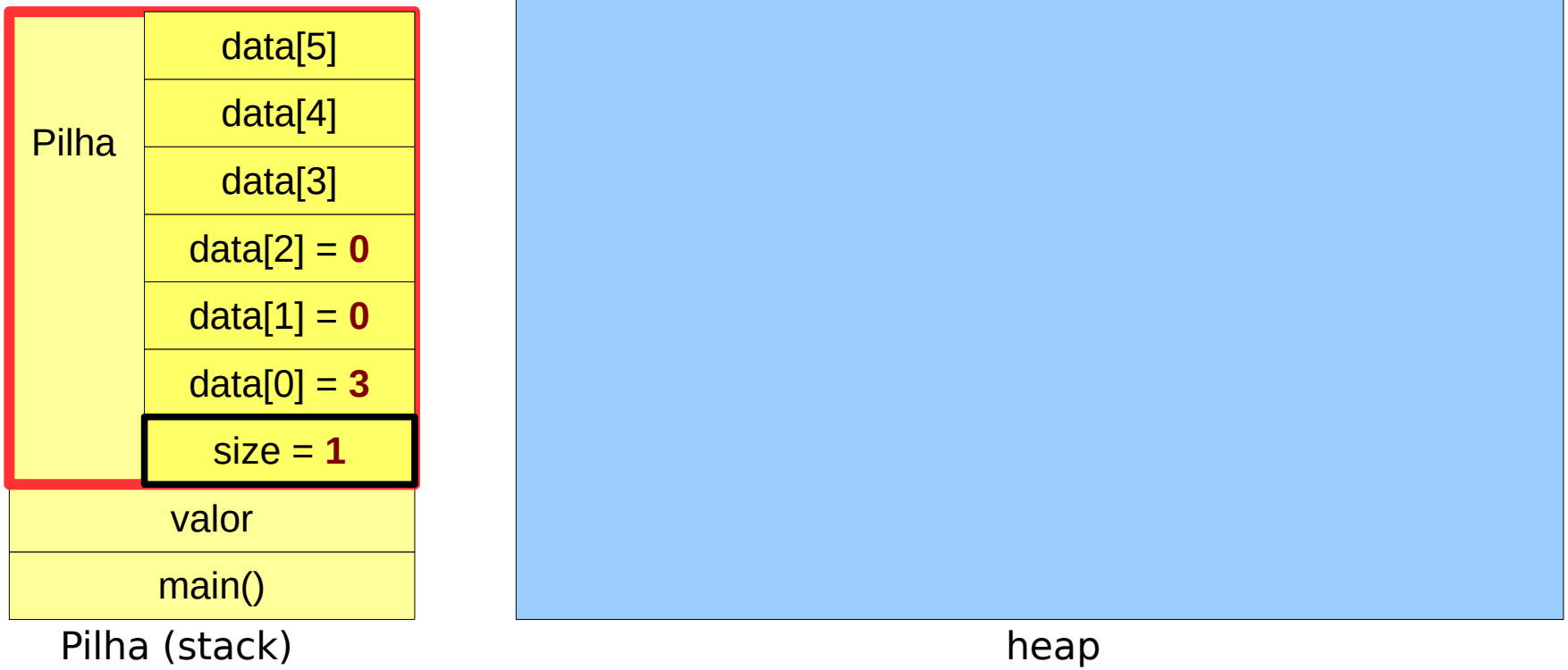
pilha1.c
Variáveis



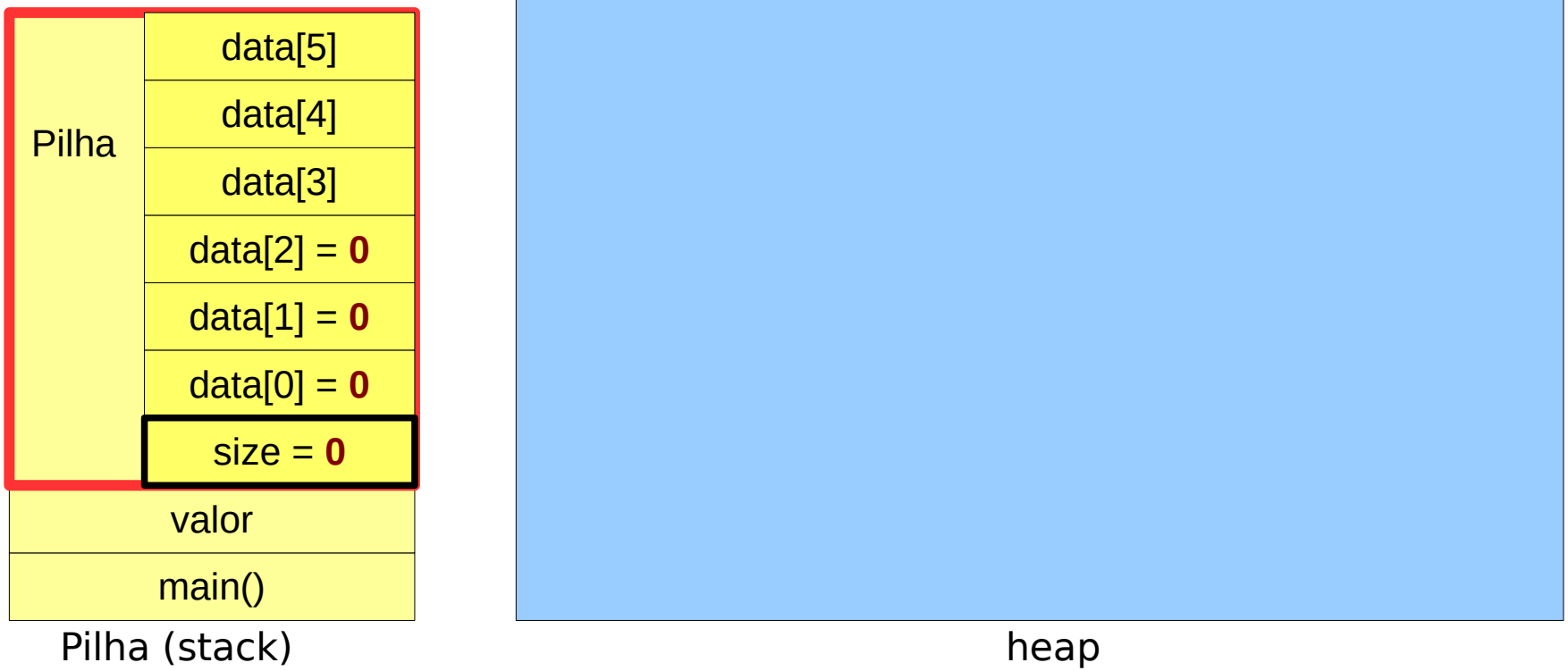
pilha1.c
Variáveis



pilha1.c
Variáveis



pilha1.c
Variáveis



Implementando pilha com C
Segunda versão

Prof. Me. Orlando Saraiva Júnior

Segunda versão

Nesta segunda versão, não há um limite para o número de elementos que podem ser empilhados.

Uma estrutura (Node) serve para armazenar o valor empilhado. Uma outra estrutura (Pilha) contém um ponteiro para o nó (Node) e o tamanho atual da pilha.

Diferente da versão anterior, nesta implementação não há um limite de elementos possíveis.

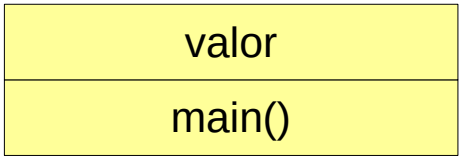
Ao empilhar um elemento (operação push), instancia um novo nó, empilhando-o na estrutura Pilha.

Ao desempilhar um elemento (operação pop), o topo da pilha é desempilhado e o nó no topo é desvinculado (via chamada de sistema free()).

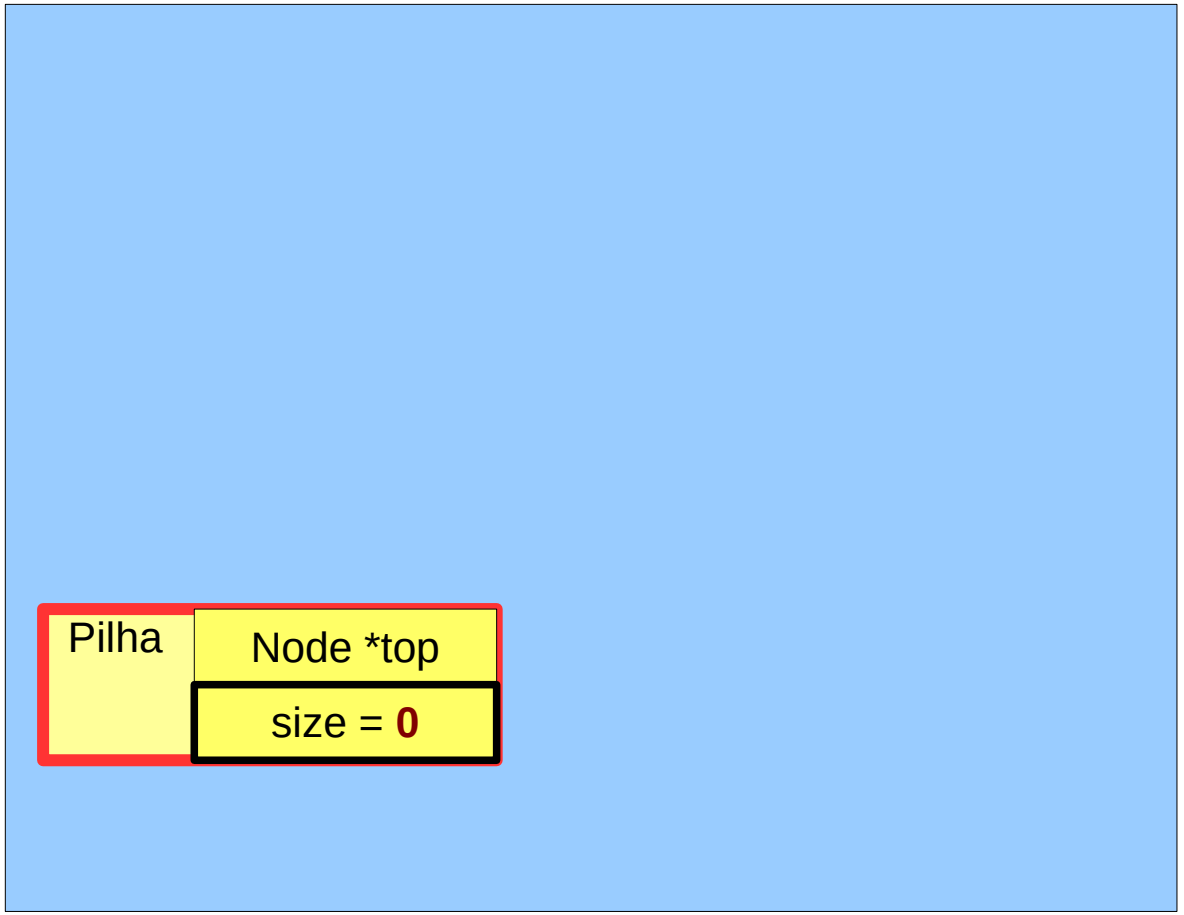
pilha2.c

Nenhum nó alocado (ainda)

Linha 86: A pilha é instanciada, mas nenhum nó, por enquanto.



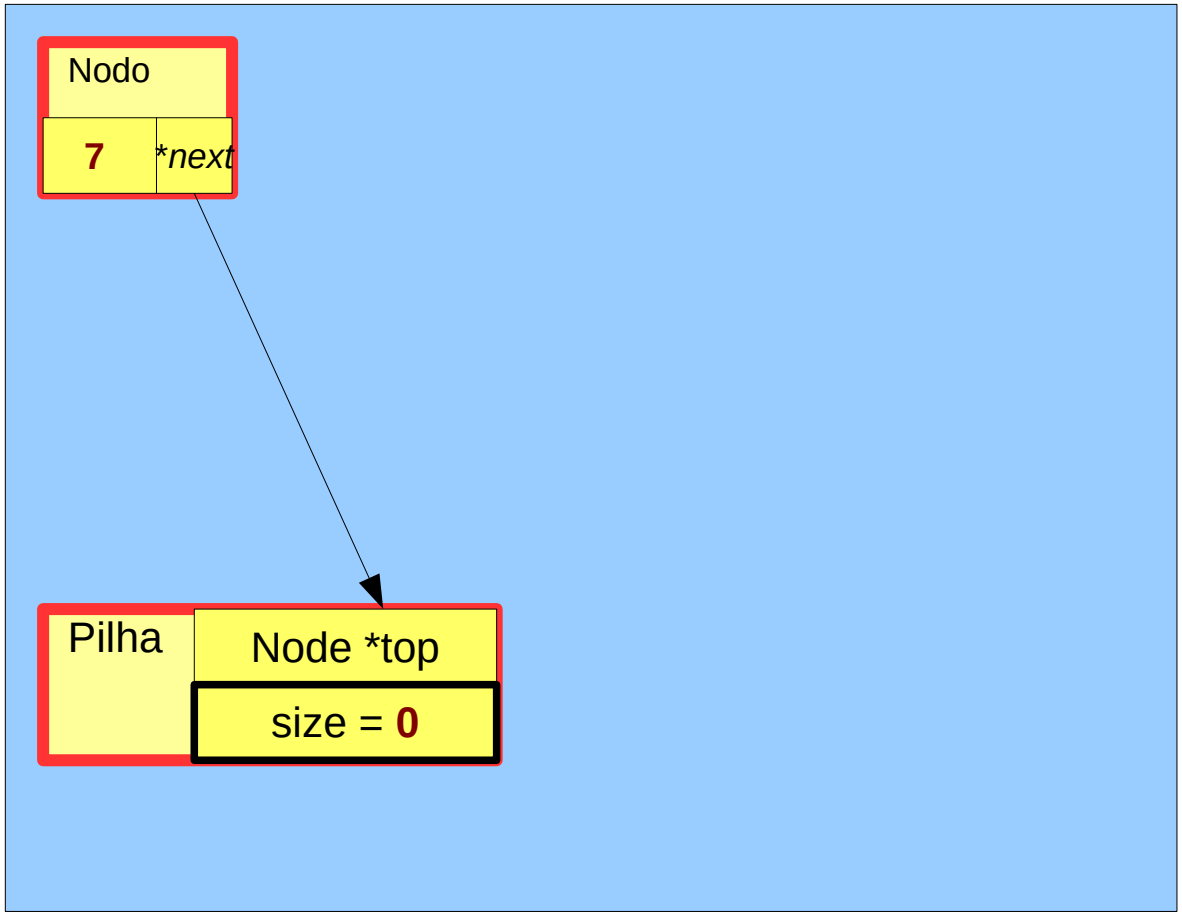
Pilha (stack)

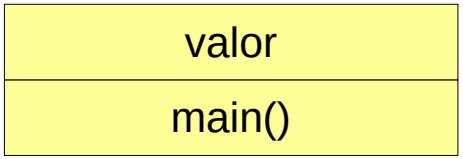


heap

pilha2.c
Primeiro push

Linha 36





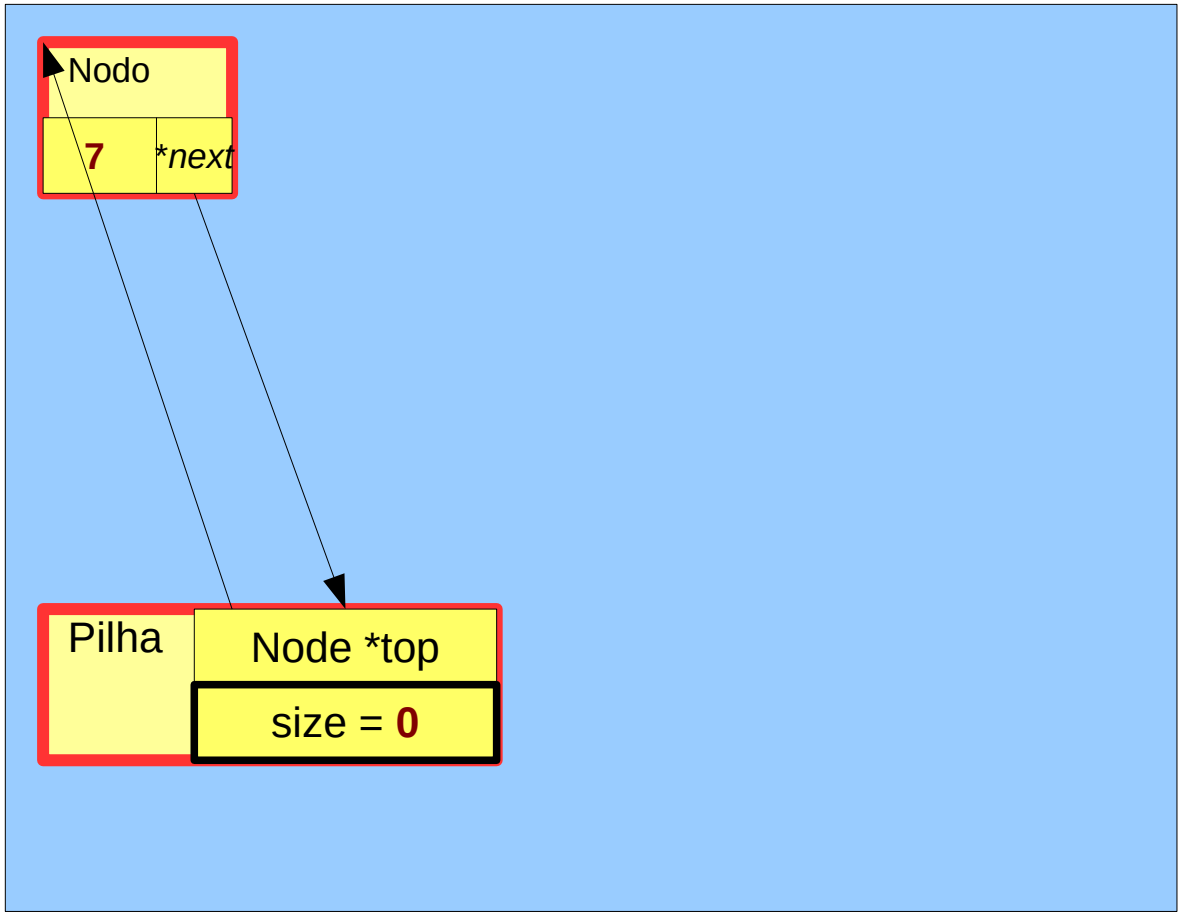
Pilha (stack)

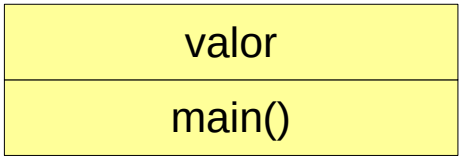
heap

pilha2.c
Primeiro push

Linha 37

O contador size será incrementado na linha 39

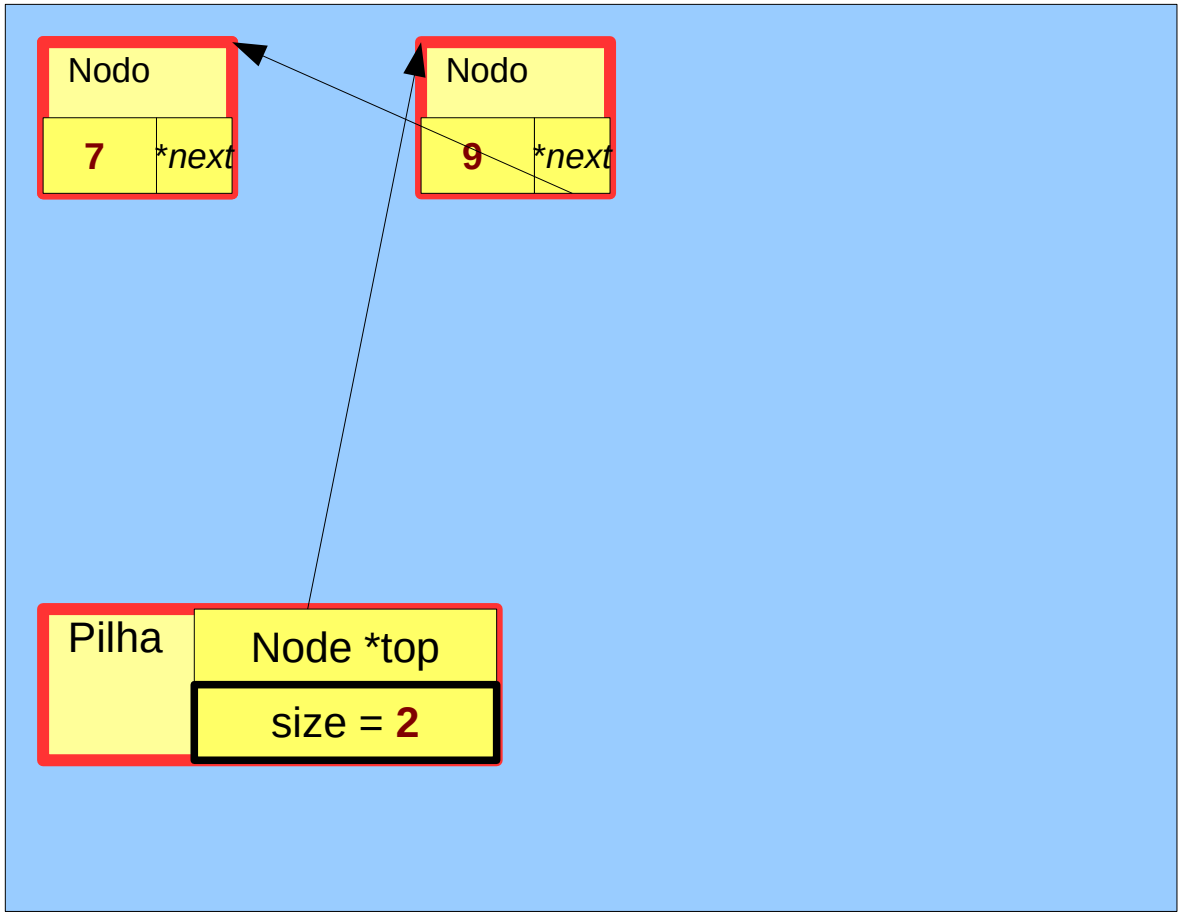


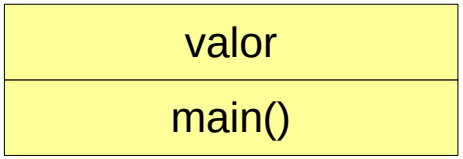


Pilha (stack)

heap

pilha2.c
Segundo push





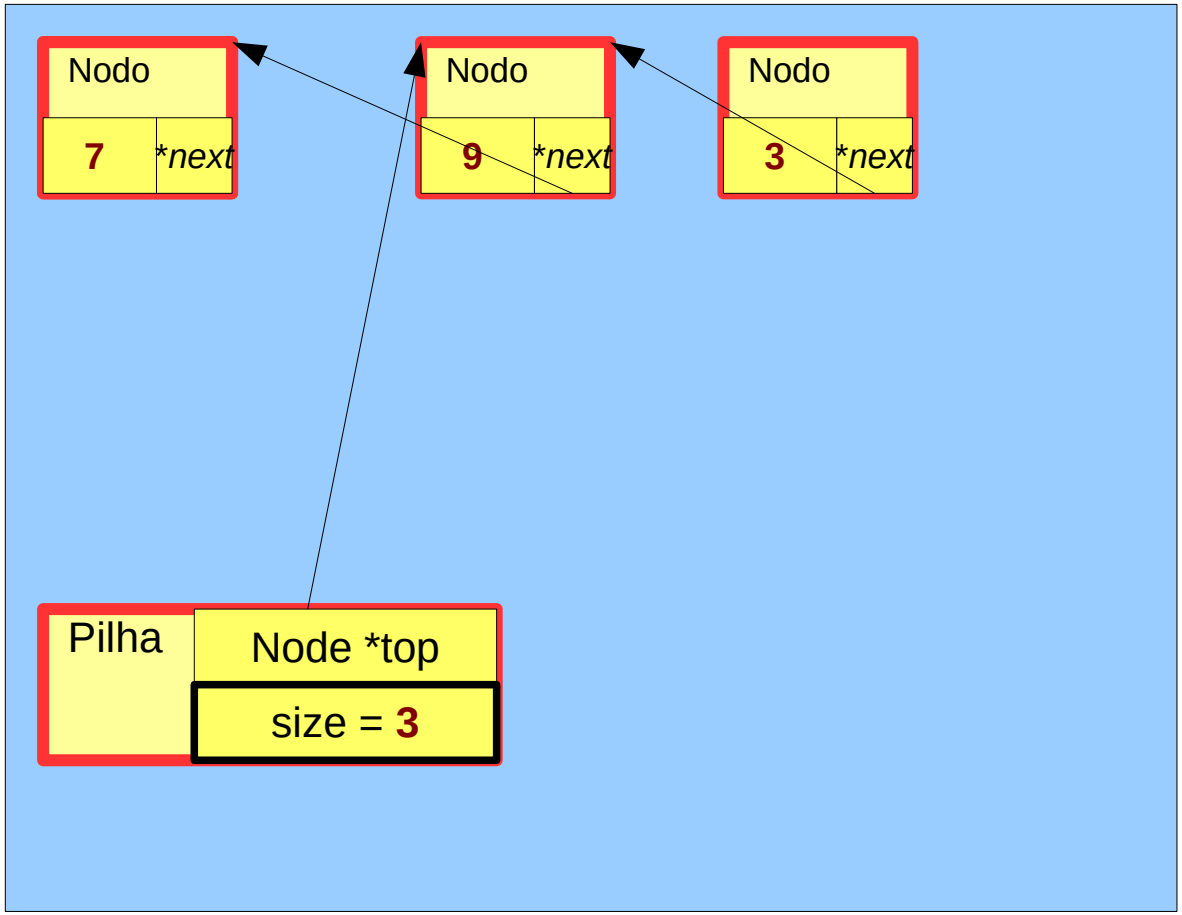
Pilha (stack)

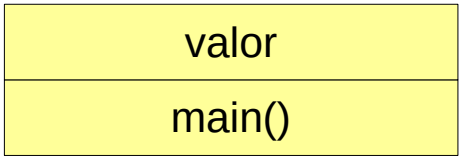
heap

pilha2.c

Terceiro push

Linha 36





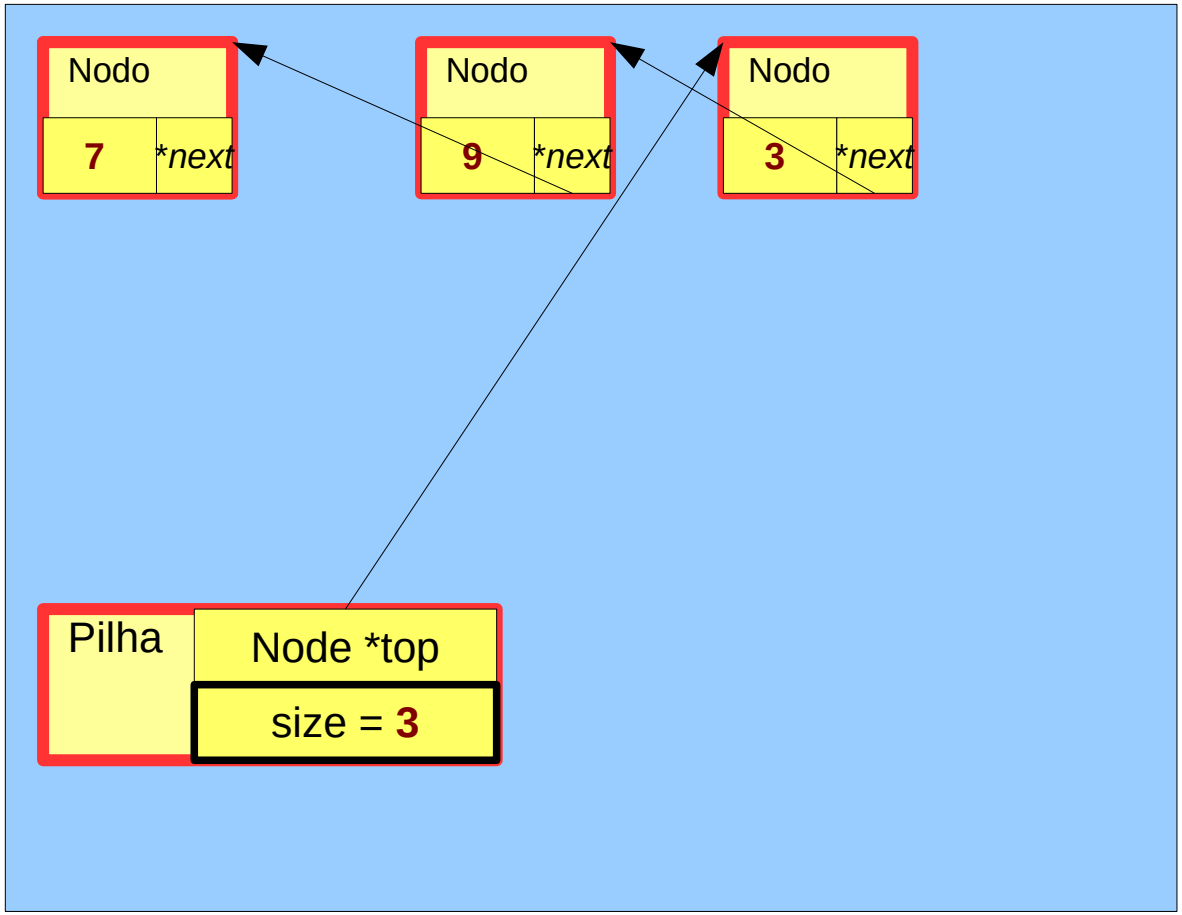
Pilha (stack)

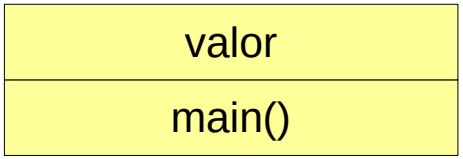
heap

pilha2.c

Terceiro push

Linha 37



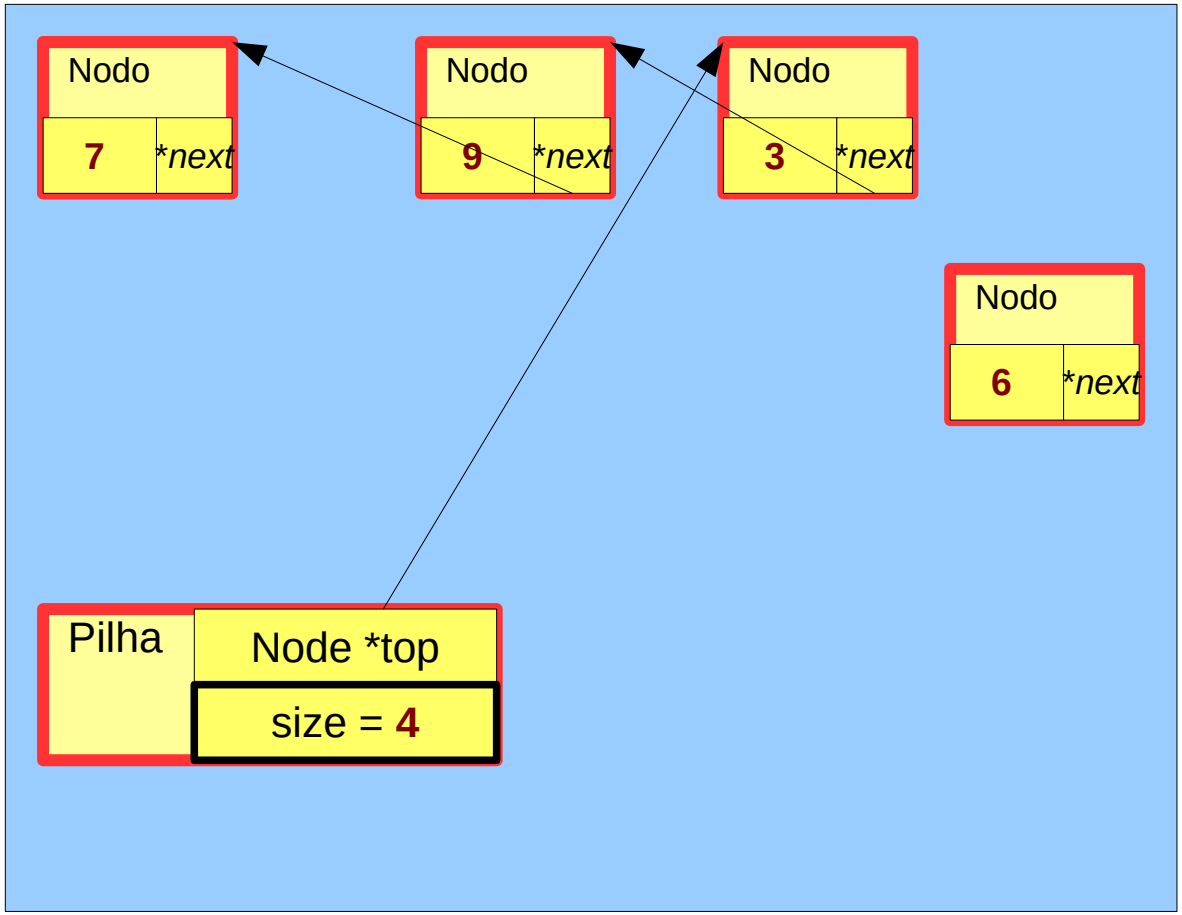


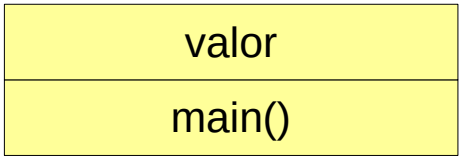
Pilha (stack)

heap

pilha2.c
Quarto push

Linha 35





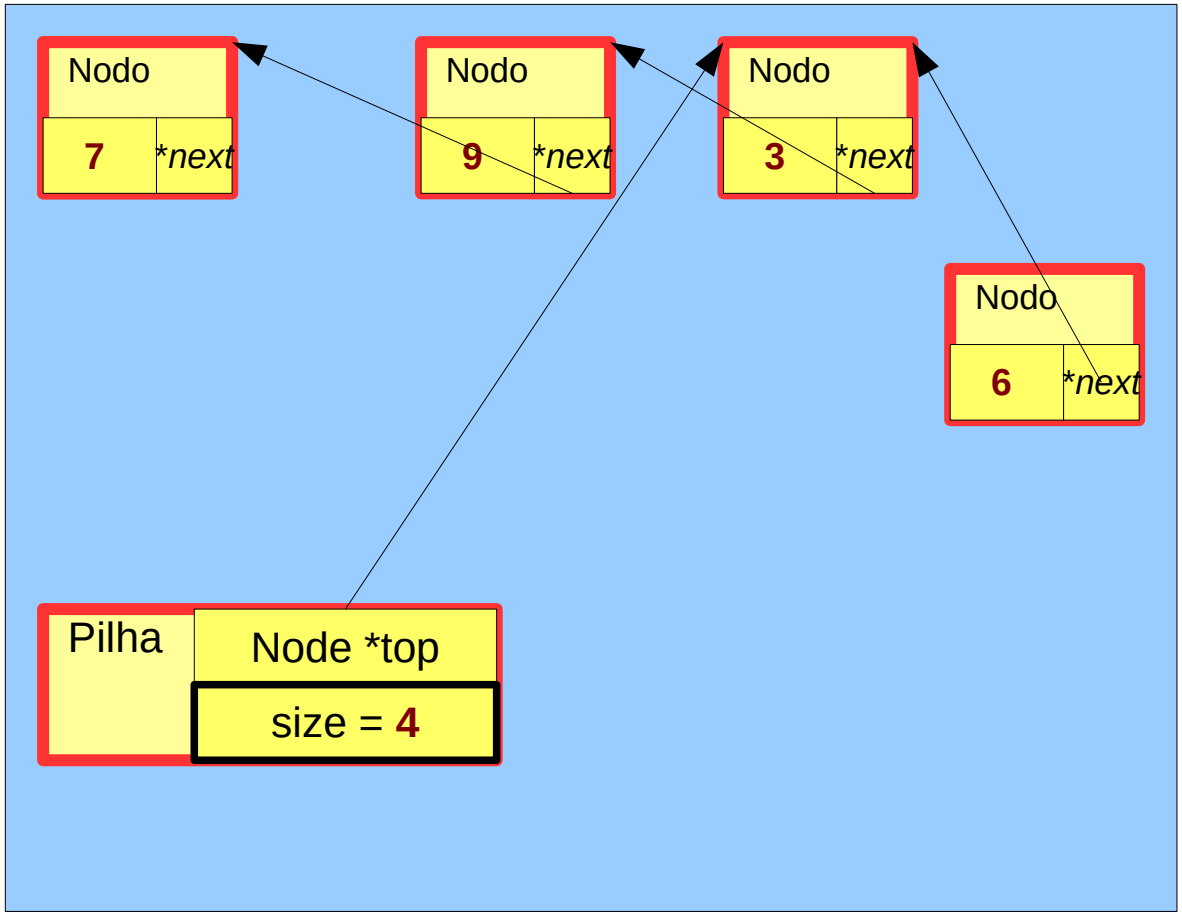
Pilha (stack)

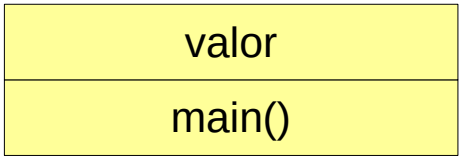
heap

pilha2.c

Quarto push

Linha 36





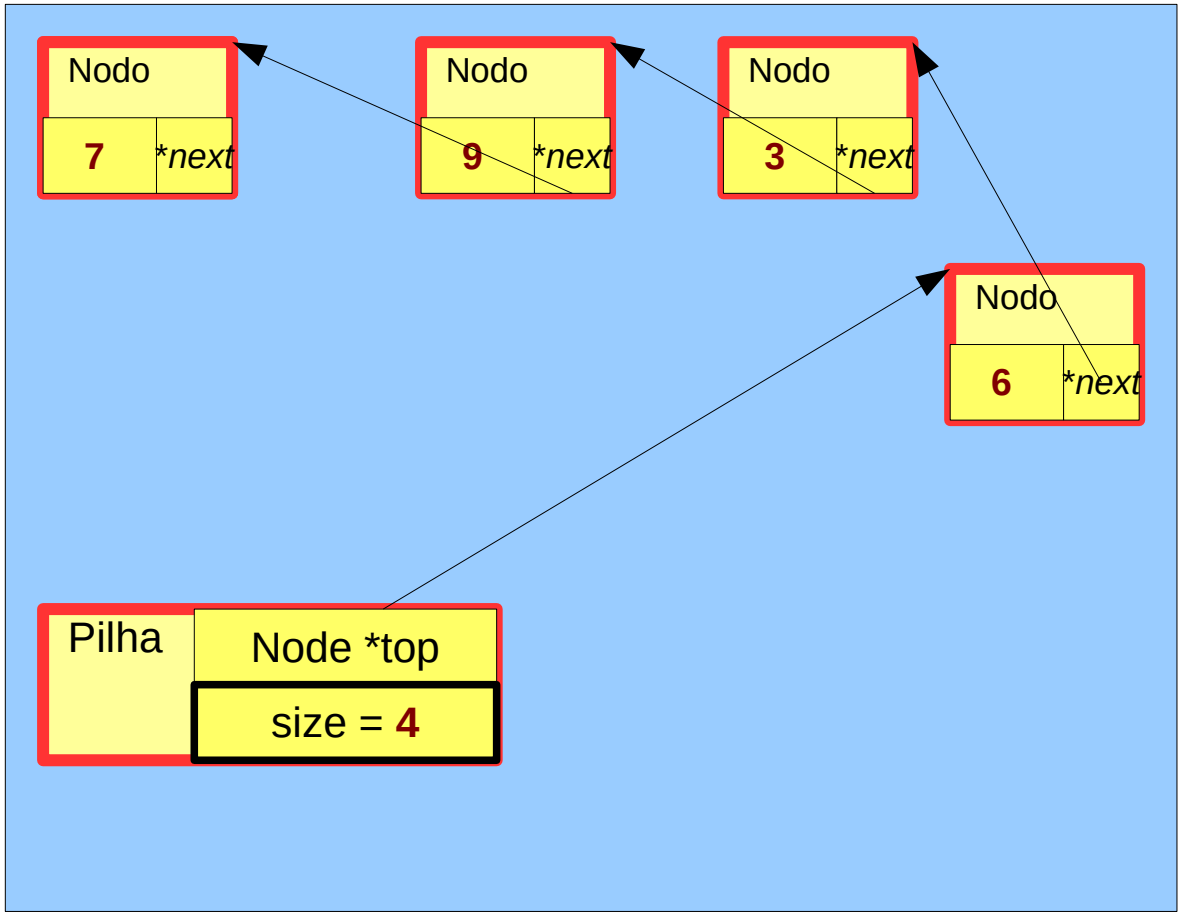
Pilha (stack)

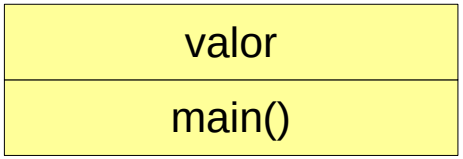
heap

pilha2.c

Quarto push

Linha 37

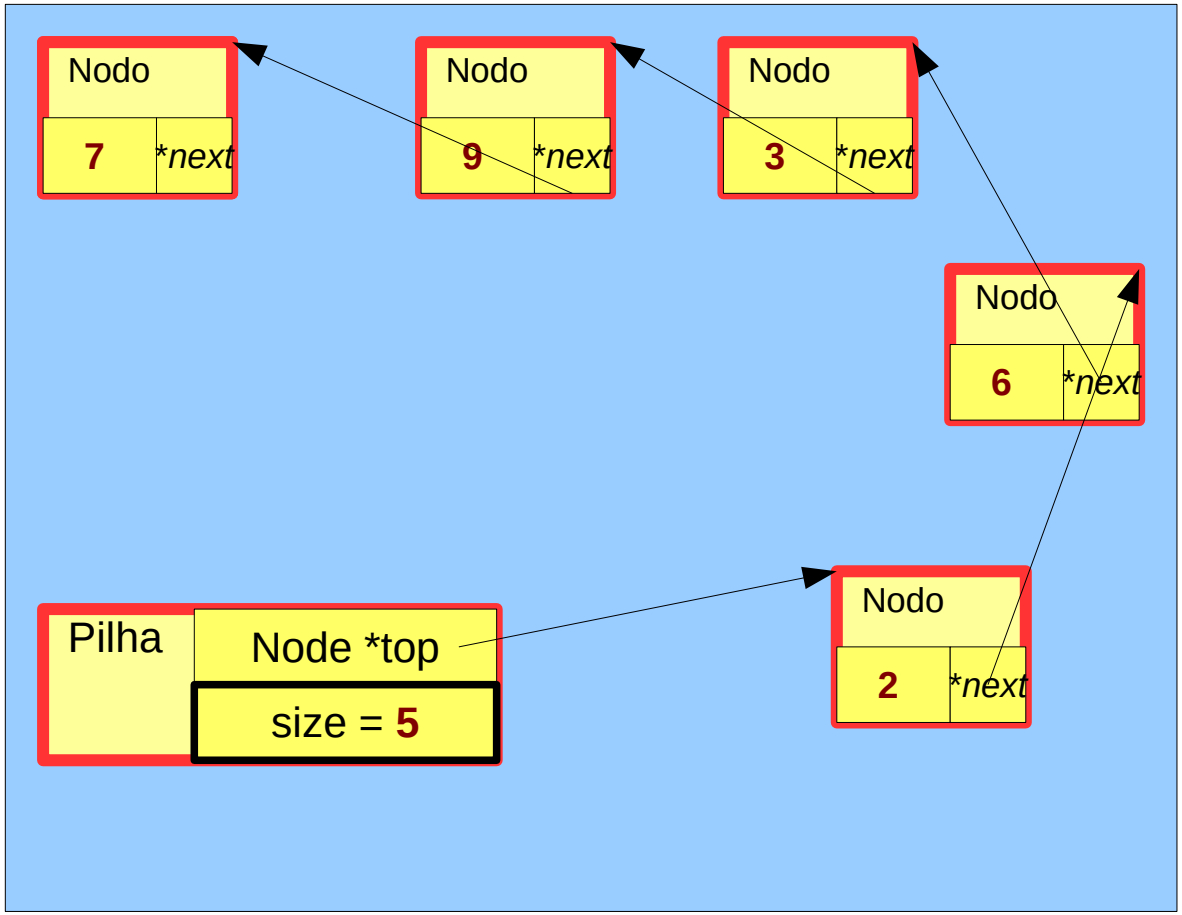


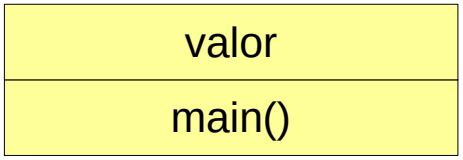


Pilha (stack)

heap

pilha2.c
Quinto push



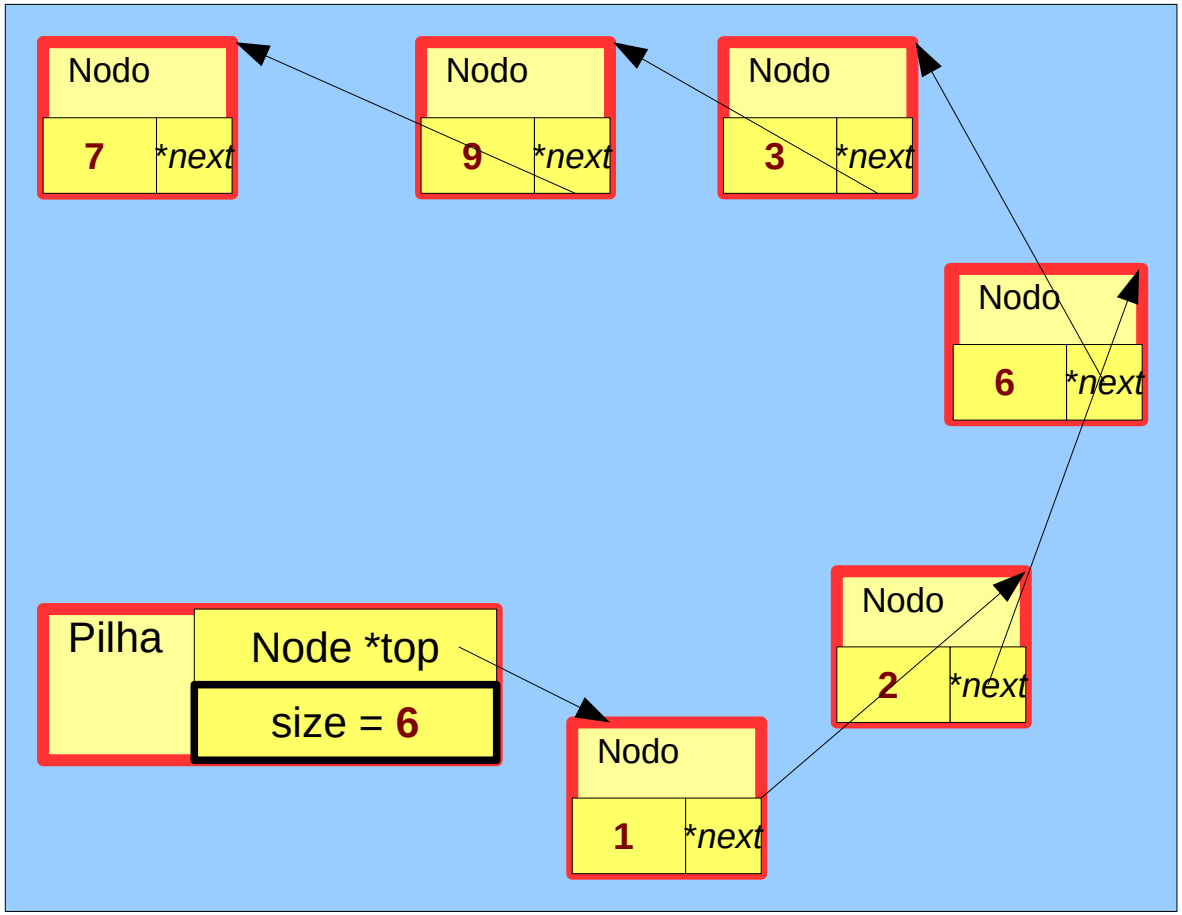


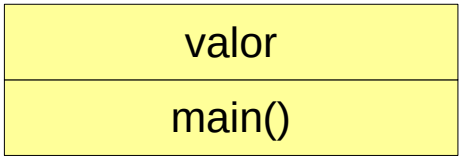
Pilha (stack)

heap

pilha2.c

Sexto push

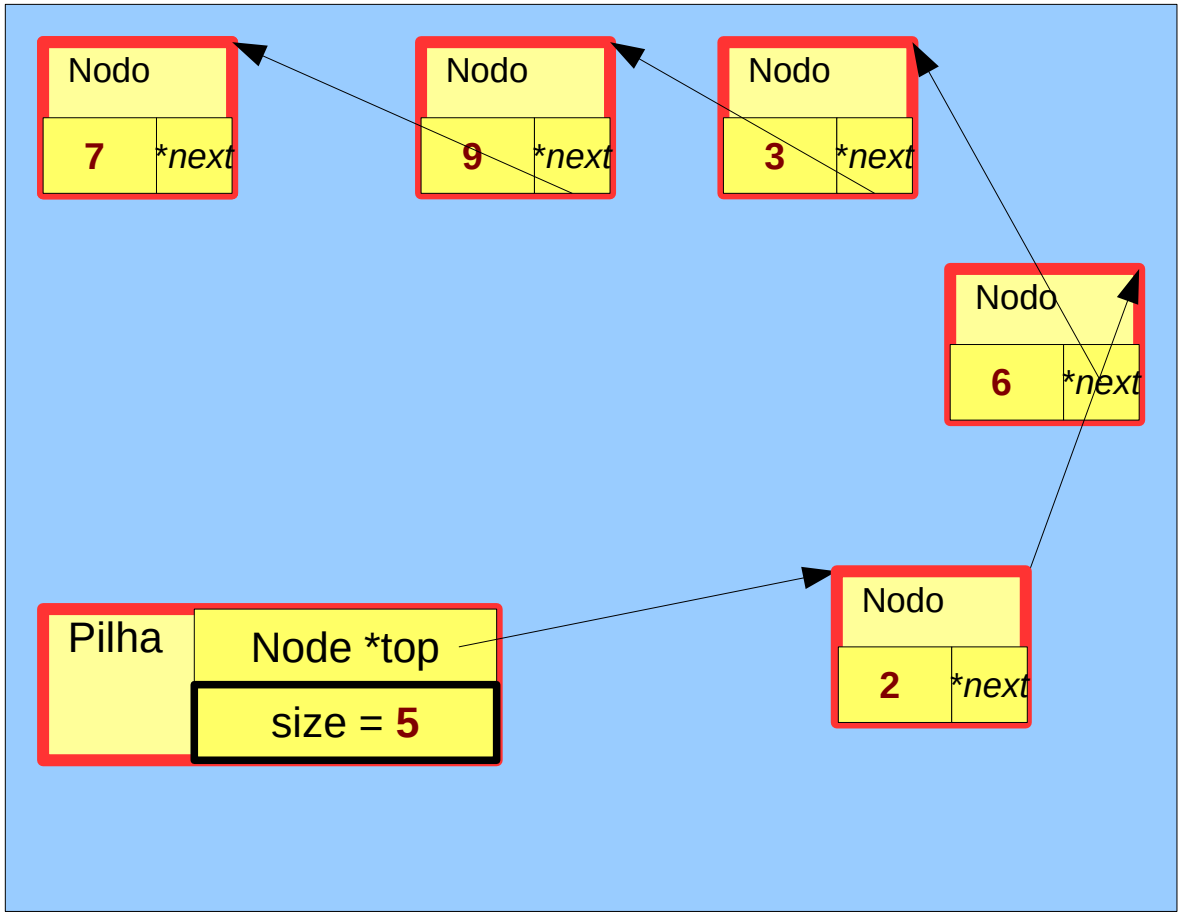


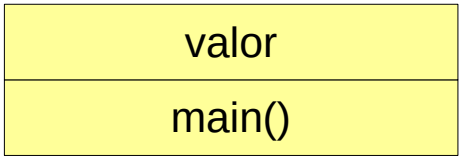


Pilha (stack)

heap

pilha2.c
Primeiro POP



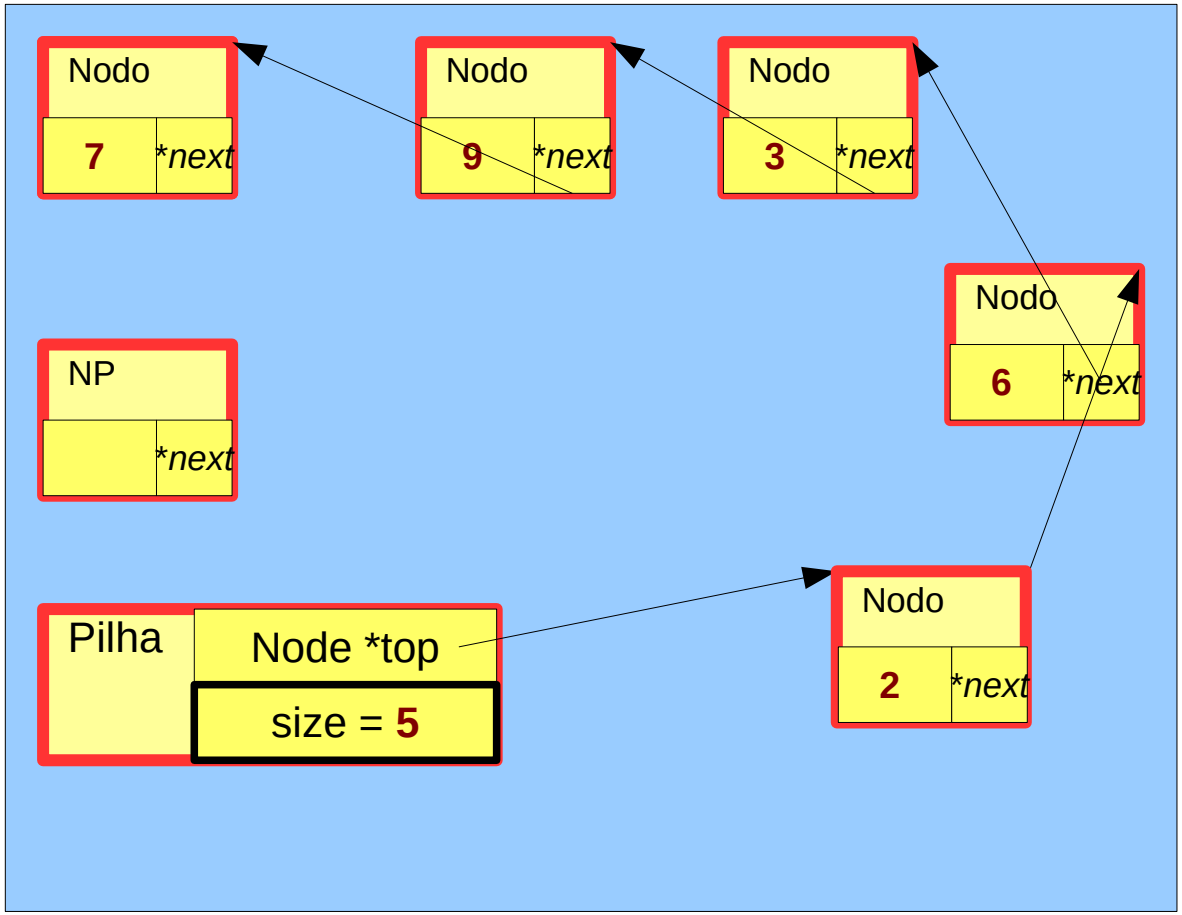


Pilha (stack)

heap

pilha2.c
Segundo POP

Linha 44



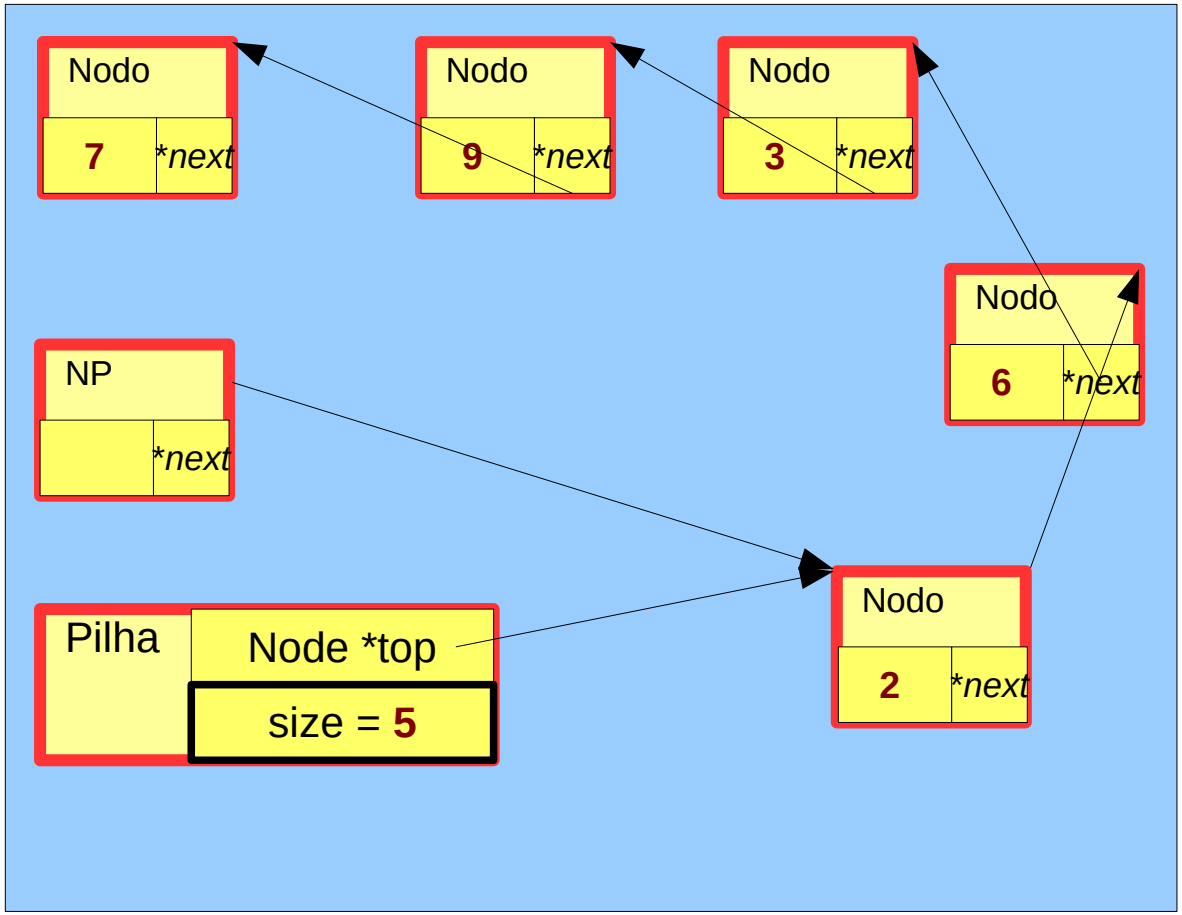
temp
valor
main()

Pilha (stack)

heap

pilha2.c
Segundo POP

Linha 51



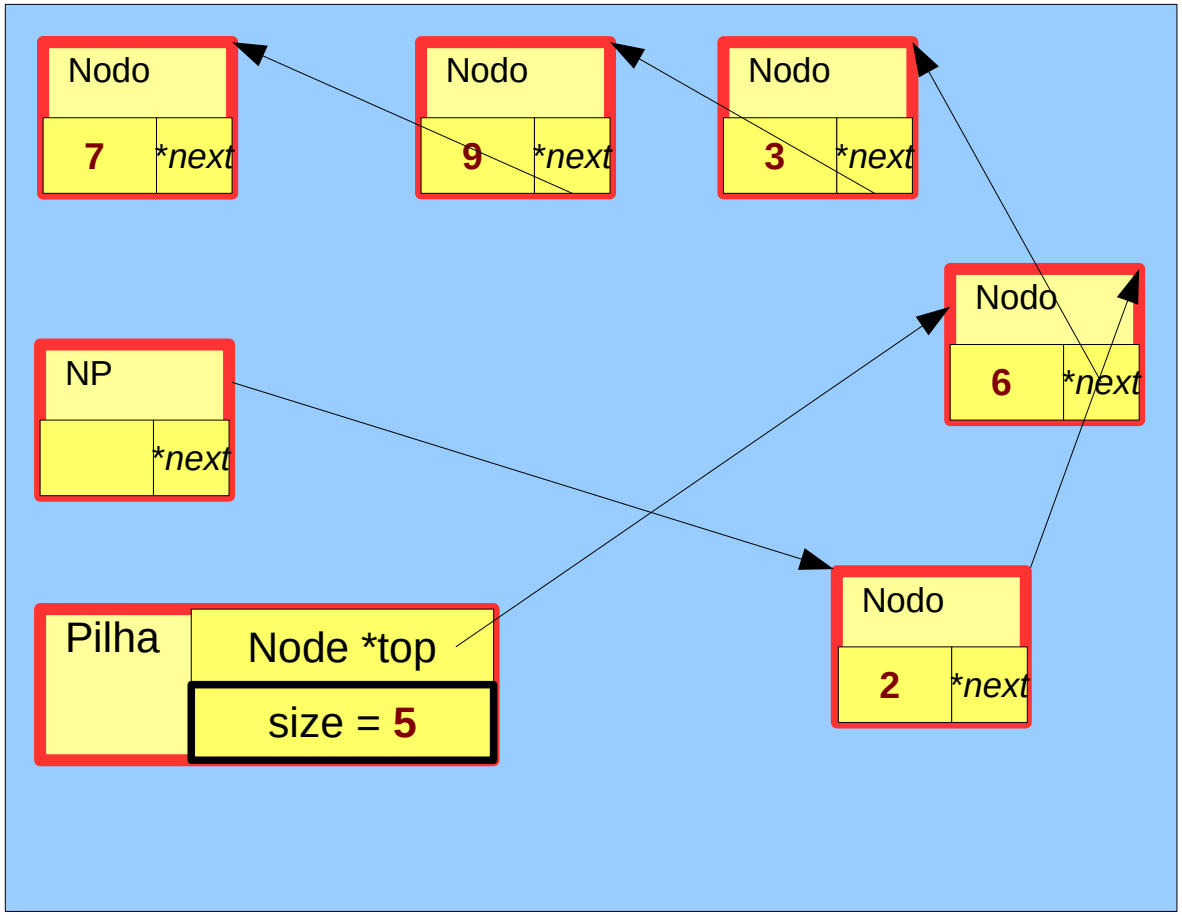
temp = **2**
valor
main()

Pilha (stack)

heap

pilha2.c
Segundo POP

Linha 52



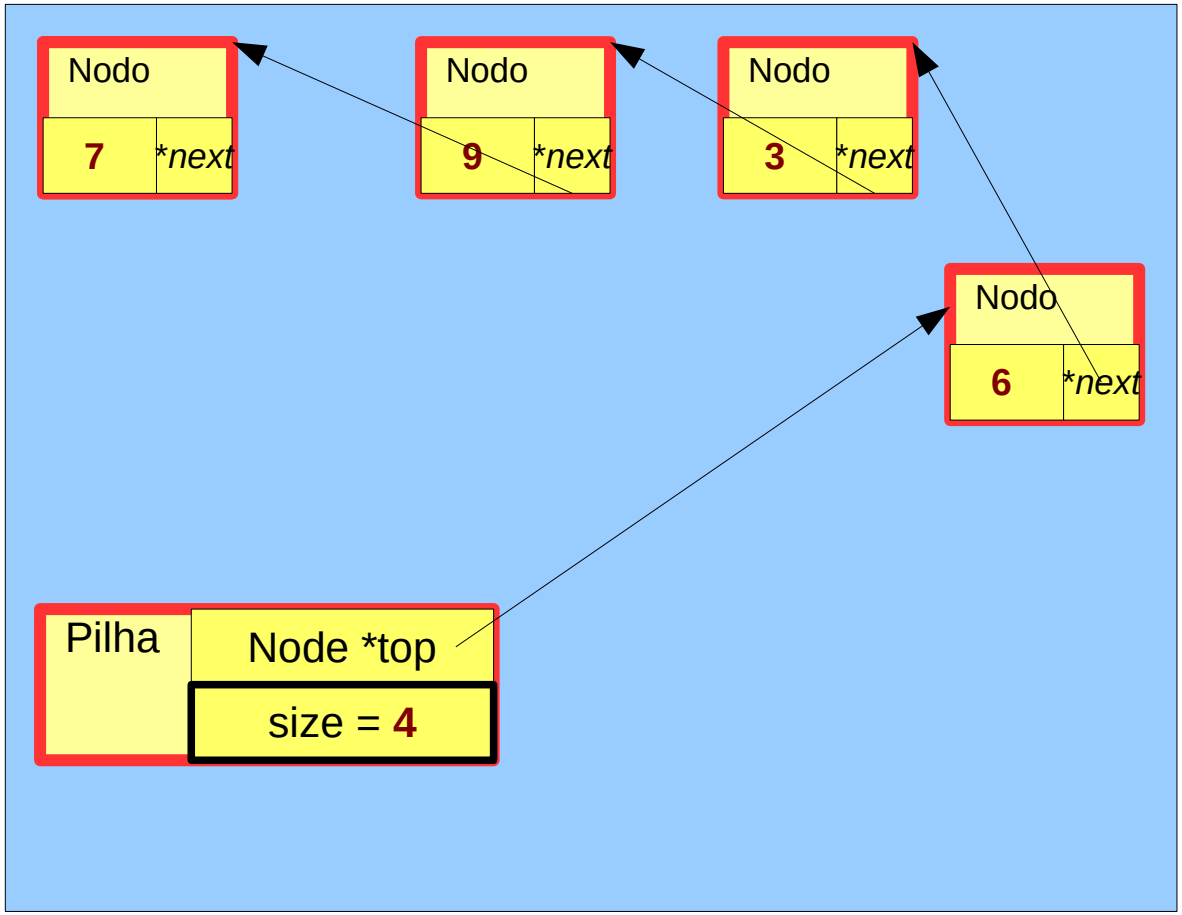
temp = **2**
valor
main()

Pilha (stack)

heap

pilha2.c
Segundo POP

Linha 54



temp = **2**
valor
main()

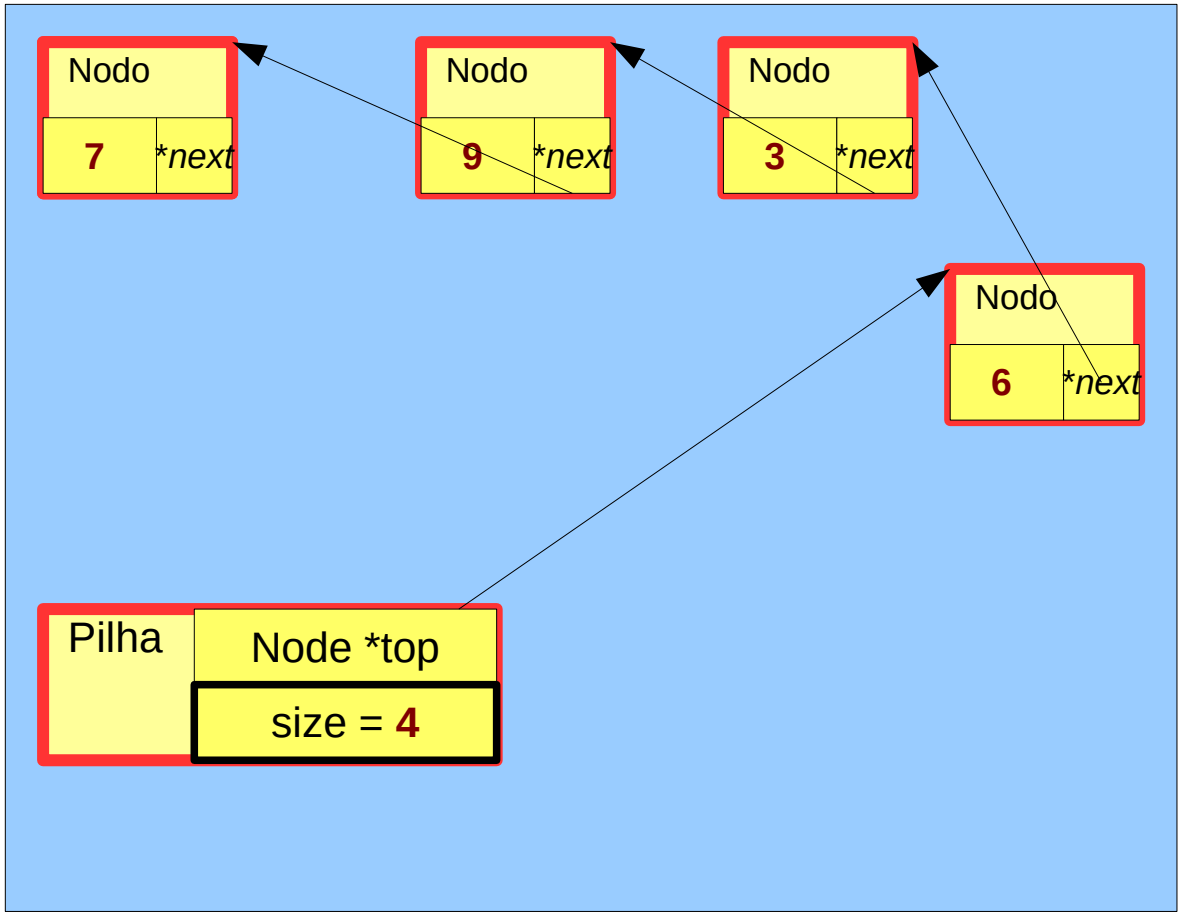
Pilha (stack)

heap

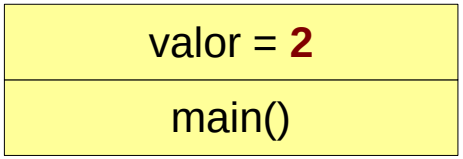
pilha2.c

Returno do segundo POP

Linha 99
O retorno da função pop invocado no *main* retorna o valor da variável *temp*



heap



Pilha (stack)

Fila

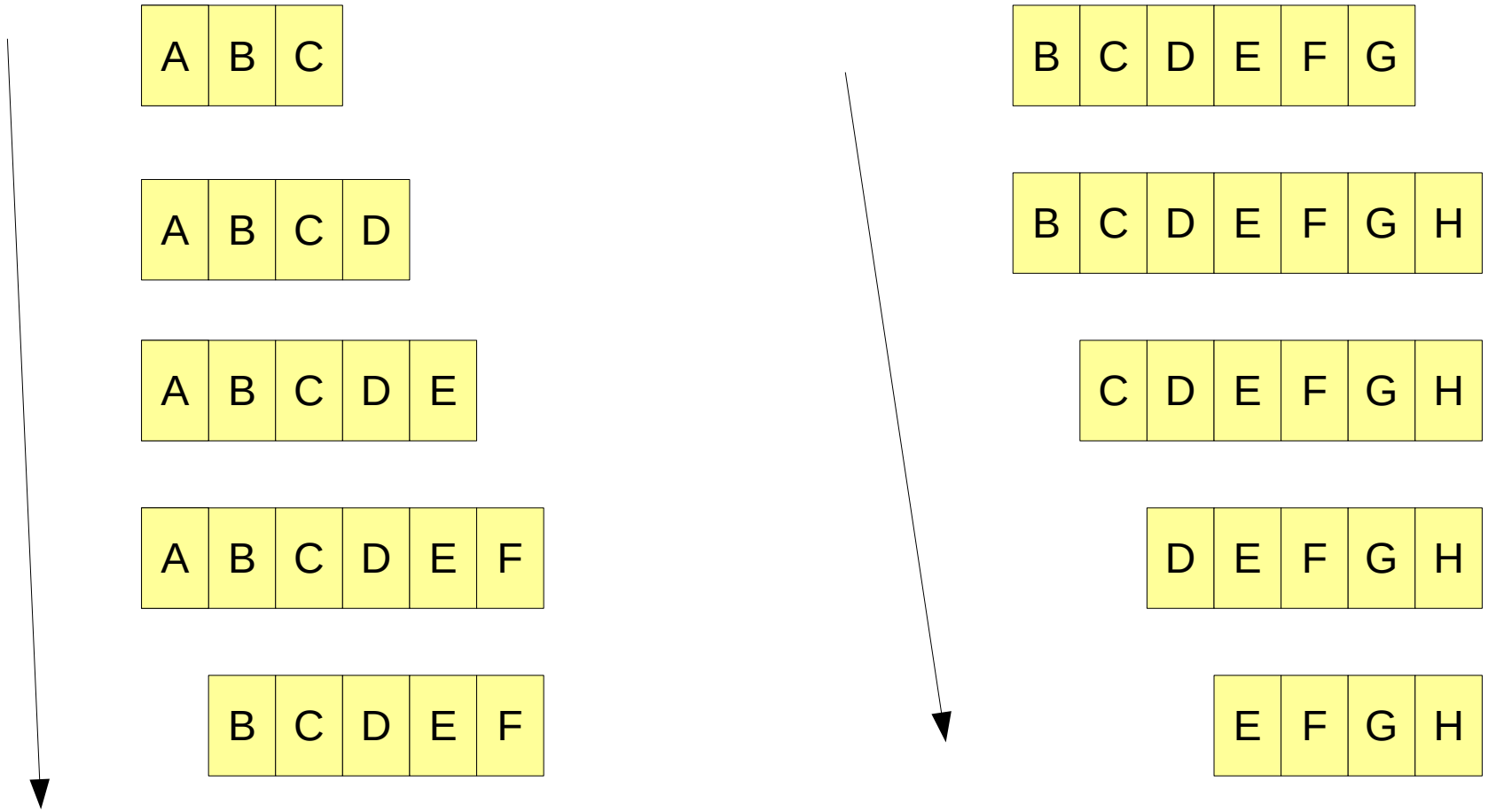
Fila

Uma **fila** é um conjunto ordenado de itens a partir do qual podem-se eliminar itens numa extremidade (chamado de **início** da fila) e no qual podem-se inserir itens na outra extremidade (chamada **final** da fila)

Possuem duas funções básicas: ENQUEUE, que adiciona um elemento ao final da fila, e DEQUEUE, que remove o elemento no início da fila.

O primeiro item a ser inserido na pilha é o primeiro a ser removido. Esta política é conhecida pela sigla FIFO (**F**irst **I**n **F**irst **O**ut), ao contrário da pilha.

Fila


A B C
A B C D
A B C D E
A B C D E F
B C D E F
B C D E F G
B C D E F G H
C D E F G H
D E F G H
E F G H

Aplicação

Exemplo 1

O escalonamento dos processos em um sistema operacional é feito por filas. O escalonador do sistema operacional utiliza uma fila de processos, dando um tempo t de CPU para cada um.

Exemplo 2

O controle de estoque serve para a empresa avaliar a entrada e saída de mercadorias e auxilia uma companhia a reduzir custos e administrar a cadeia de produção e distribuição com mais eficiência.

Filas

Formalmente, uma fila é um tipo de dado abstrato (ADT) tal que uma instância Q (de *queue*) suporta as seguintes funcionalidades:

Q.enqueue(elem) → Adiciona o elemento elem ao fim da fila Q

Q.dequeue() → Remove e retorna o primeiro elemento da fila Q.

Espera-se um erro caso a fila esteja vazia.

Filas

No exemplo, teremos outros três funcionalidades

Q.first() → Retorna a referência ao elementono início da fila.

Q.is_empty() → Returna verdadeiro, caso a fila esteja vazia.

Q.tamanho() → Returna o número de elementos da fila Q.

Filas

Operação	Valor de retorno	Conteúdo da fila
Q.enqueue(5)	-	[5]
Q.enqueue(3)	-	[5, 3]
Q.tamanho()	2	[5, 3]
Q.dequeue()	5	[3]
Q.is_empty()	False	[3]
Q.pop()	5	[ ]
Q.is_empty()	True	[ ]
Q.dequeue()	“erro”	[ ]
Q.enqueue(7)	-	[7]
Q.enqueue(9)	-	[7, 9]
Q.enqueue(4)	-	[7, 9 , 4]
Q.tamanho()	3	[7, 9 , 4]
Q.enqueue(8)	-	[7, 9 , 4 , 8]

Filas

Operação	Valor de retorno	Conteúdo da fila
Q.enqueue(6)	-	[7 , 9 , 4 , 8 , 6]
Q.enqueue(1)	? ? ?	? ? ?
Q.tamanho()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.is_empty()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.enqueue(1)	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?
Q.dequeue()	? ? ?	? ? ?

Filas

Operação	Valor de retorno	Conteúdo da fila
Q.enqueue(6)	-	[7 , 9 , 4 , 8 , 6]
Q.enqueue(1)	-	[7 , 9 , 4 , 8 , 6 , 1]
Q.tamanho()	6	[7 , 9 , 4 , 8 , 6 , 1]
Q.dequeue()	7	[9 , 4 , 8 , 6 , 1]
Q.is_empty()	False	[9 , 4 , 8 , 6 , 1]
Q.dequeue()	9	[4 , 8 , 6 , 1]
Q.dequeue()	4	[8 , 6 , 1]
Q.dequeue()	8	[6 , 1]
Q.dequeue()	6	[1]
Q.enqueue(1)	-	[1, 1]
Q.dequeue()	1	[1]
Q.dequeue()	1	[ ]
Q.dequeue()	“erro”	[ ]

Implementando fila com C
Primeira versão

Prof. Me. Orlando Saraiva Júnior

Fila em C: Primeira versão

A estrutura Fila contém um array de inteiros data de tamanho QUEUE_MAX.

Dois inteiros inicio e fim para rastrear o início e o fim da fila, e um inteiro size que armazena o número de elementos na fila.

Implementando fila com C
Segunda versão

Prof. Me. Orlando Saraiva Júnior

Pilha em C: Segunda versão

Nesta segunda versão, **não há um limite** para o número de elementos que podem ser enfileirados.

Como não há um limite fixo, a fila pode crescer dinamicamente conforme necessário, até o limite de memória disponível.

Síntese

Dúvidas

Prof. Orlando Saraiva do Nascimento Júnior
orlando.saraiva@unesp.br
orlando.nascimento@fatec.sp.gov.br