Guilherme dos Santos Ferreira

Estado de Memória - Atividade B1-3 - Manipulação de Lista Ligada

Inserir Pedido

Python Tutor: Visualize Code and Get AI Help for Python, JavaScript, C, C++, and Java

C (C17 + GNU extensions)
known limitations

```
75      struct Pedido *atual = *inicio;
76      struct Pedido *prox;
77      while (atual != NULL) {
78          prox = atual->prox;
79          free(atual);
80          atual = prox;
81      }
82      *inicio = NULL;
83      printf("Lista de pedidos liberada!\n");
84  }
85  int main() {
86      struct Pedido *inicio = criarLista();
87      inserirPedido(&inicio, 101, "João", "Pizza", 2, '
88      inserirPedido(&inicio, 102, "Maria", "Hamburguer'
89      obterPedido(inicio, 102);
90      atualizarStatus(inicio, 101, "entregue");
91      deletarPedido(&inicio, 102);
92      liberarLista(&inicio);
93      return 0;
94  }
```

Edit this code

→ line that just executed
→ next line to execute

Print output (drag lower right corner to resize)

Stack   Heap

main

| inicio | pointer to Pedido
NULL (0x0) |

C/C++ details: none [default view]

<< First   < Prev   Next >   Last >>
Step 4 of 76

C (C17 + GNU extensions)
known limitations

```
 4  struct Pedido {
 5      int num;
 6      char nomeCliente[30];
 7      char prato[30];
 8      int quant;
 9      char status[10];
10      struct Pedido *prox;
11  };
12  struct Pedido* criarLista() {
13      return NULL;
14  }
→15  void inserirPedido(struct Pedido **inicio, int num, (
16      struct Pedido *novoPedido = (struct Pedido*)mall(
17      if (novoPedido == NULL) {
18          printf("Erro ao alocar memória!\n");
19          return;
20      }
21      novoPedido->num = num;
22      strcpy(novoPedido->nomeCliente, nomeCliente);
23      strcpy(novoPedido->prato, prato);
24      novoPedido->quant = quant;
```

Edit this code

→ line that just executed
→ next line to execute

[<< First] [< Prev] [Next >] [Last >>]

Step 5 of 76

follow our YouTube, TikTok, Instagram for weekly tutorials

Move and hide objects

Print output (drag lower right corner to resize)

Stack   Heap

main

inicio   pointer to Pedido
         NULL (0x0)

inserirPedido

inicio   pointer to Pedido*
         ?

num   int
         ?

nomeCliente   pointer to char
              ?

prato   pointer to char
         ?

quant   int
         ?

status   pointer to char
         ?

novoPedido   pointer to Pedido
             ?

Note: ? refers to an uninitialized value

C/C++ details: none [default view]

C (C17 + GNU extensions)
known limitations

```
 4  struct Pedido {
 5      int num;
 6      char nomeCliente[30];
 7      char prato[30];
 8      int quant;
 9      char status[10];
10      struct Pedido *prox;
11  };
12  struct Pedido* criarLista() {
13      return NULL;
14  }
→15  void inserirPedido(struct Pedido **inicio, int num, (
→16     struct Pedido *novoPedido = (struct Pedido*)mall(
17      if (novoPedido == NULL) {
18          printf("Erro ao alocar memória!\n");
19          return;
20      }
21      novoPedido->num = num;
22      strcpy(novoPedido->nomeCliente, nomeCliente);
23      strcpy(novoPedido->prato, prato);
24      novoPedido->quant = quant;
```

Edit this code

→ line that just executed
→ next line to execute

[<< First] [< Prev] [Next >] [Last >>]

Step 6 of 76

follow our YouTube, TikTok, Instagram for weekly tutorials

Move and hide objects

Print output (drag lower right corner to resize)

Stack   Heap

main

inicio   pointer to Pedido
         NULL (0x0)

array
char	char	char	char	char	char
0	1	2	3	4	5
'J'	'o'	'\xC3'	'\xA3'	'o'	'\0'
(this is in read-only storage, not the heap)

inserirPedido

inicio   pointer to Pedido*

num   int
         101

nomeCliente   pointer to char

prato   pointer to char

quant   int
         2

status   pointer to char

novoPedido   pointer to Pedido
             ?

array
char	char	char	char	char	char
0	1	2	3	4	5
'P'	'i'	'z'	'z'	'a'	'\0'
(this is in read-only storage, not the heap)

array
char	char	char	char	char	char	char	char	char
0	1	2	3	4	5	6	7	8
'p'	'e'	'n'	'd'	'e'	'n'	't'	'e'	'\0'
(this is in read-only storage, not the heap)

Note: ? refers to an uninitialized value

C/C++ details: none [default view]

C (C17 + GNU extensions)
known limitations

```c
 15  void inserirPedido(struct Pedido **inicio, int num, (
 16      struct Pedido *novoPedido = (struct Pedido*)mall
 17      if (novoPedido == NULL) {
 18          printf("Erro ao alocar memória!\n");
 19          return;
 20      }
 21      novoPedido->num = num;
 22      strcpy(novoPedido->nomeCliente, nomeCliente);
 23      strcpy(novoPedido->prato, prato);
 24      novoPedido->quant = quant;
 25      strcpy(novoPedido->status, status);
 26      novoPedido->prox = *inicio;
 27      *inicio = novoPedido;
 28
 29      printf("Pedido #%d inserido!\n", num);
 30  }
 31  void obterPedido(struct Pedido *inicio, int num) {
 32      struct Pedido *ptr = inicio;
 33      while (ptr != NULL) {
```

Edit this code

→ line that just executed
⇒ next line to execute

<< First   < Prev   Next >   Last >>
Step 16 of 76

follow our YouTube, TikTok, Instagram for weekly tutorials

Move and hide objects

Print output (drag lower right corner to resize)
Pedido #101 inserido!

Stack   Heap

C (C17 + GNU extensions)
known limitations

```c
 75      struct Pedido *atual = inicio;
 76      struct Pedido *prox;
 77      while (atual != NULL) {
 78          prox = atual->prox;
 79          free(atual);
 80          atual = prox;
 81      }
 82      *inicio = NULL;
 83      printf("Lista de pedidos liberada!\n");
 84  }
 85  int main() {
 86      struct Pedido *inicio = criarLista();
 87      inserirPedido(&inicio, 101, "João", "Pizza", 2, '
 88      inserirPedido(&inicio, 102, "Maria", "Hamburguer'
 89      obterPedido(inicio, 102);
 90      atualizarStatus(inicio, 101, "entregue");
 91      deletarPedido(&inicio, 102);
 92      liberarLista(&inicio);
 93      return 0;
 94  }
```

Edit this code

→ line that just executed
⇒ next line to execute

<< First   < Prev   Next >   Last >>
Step 17 of 76

Print output (drag lower right corner to resize)
Pedido #101 inserido!

Stack   Heap

C (C17 + GNU extensions)
known limitations

```c
  4  struct Pedido {
  5      int num;
  6      char nomeCliente[30];
  7      char prato[30];
  8      int quant;
  9      char status[10];
 10      struct Pedido *prox;
 11  };
 12  struct Pedido* criarLista() {
 13      return NULL;
 14  }
 15  void inserirPedido(struct Pedido **inicio, int num, (
 16      struct Pedido *novoPedido = (struct Pedido*)mall
 17      if (novoPedido == NULL) {
 18          printf("Erro ao alocar memória!\n");
 19          return;
 20      }
 21      novoPedido->num = num;
 22      strcpy(novoPedido->nomeCliente, nomeCliente);
 23      strcpy(novoPedido->prato, prato);
```

Edit this code

→ line that just executed
⇒ next line to execute

<< First   < Prev   Next >   Last >>
Step 19 of 76

follow our YouTube, TikTok, Instagram for weekly tutorials

Move and hide objects

Print output (drag lower right corner to resize)
Pedido #101 inserido!

Stack   Heap

```
17    if (novoPedido == NULL) {
18        printf("Erro ao alocar memória!\n");
19        return;
20    }
21    novoPedido->num = num;
22    strcpy(novoPedido->nomeCliente, nomeCliente);
23    strcpy(novoPedido->prato, prato);
24    novoPedido->quant = quant;
25    strcpy(novoPedido->status, status);
26    novoPedido->prox = *inicio;
27    *inicio = novoPedido;
28
29    printf("Pedido #%d inserido!\n", num);
30 }
31 void obterPedido(struct Pedido *inicio, int num) {
32     struct Pedido *ptr = inicio;
33     while (ptr != NULL) {
```

Edit this code

→ line that just executed
➡ next line to execute

<< First | < Prev | Next > | Last >>
Step 26 of 76

Move and hide objects

Stack   Heap

main
    inicio   pointer to Pedido

inserirPedido
    inicio   pointer to Pedido*
    num   int
                102
    nomeCliente pointer to char
    prato   pointer to char
    quant   int
                1
    status   pointer to char
    novoPedido pointer to Pedido

struct Pedido
    num   int
             101
    nomeCliente array
             'J' 'o' '\xC3' '\xA3' 'o' '\0' ? ...
    prato   array
             'P' 'i' 'z' 'z' 'a' '\0' ? ...
    quant   int
             2
    status   array
             'p' 'e' 'n' 'd' 'e' 'n' 't' 'e' '\0' ?
    prox   pointer to Pedido
             NULL (0x0)

array
'M' 'a' 'r' 'i' 'a' '\0'
(this is in read-only storage, not the heap)

array
'H' 'a' 'm' 'b' 'u' 'r' 'g' 'u' 'e' 'r' '\0'
(this is in read-only storage, not the heap)

array
'e' 'm' ' ' 'p' 'r' 'e' 'p' 'a' 'r' 'o' '\0'
(this is in read-only storage, not the heap)

array
struct Pedido
    num   int
             102
    nomeCliente array
             'M' 'a' 'r' 'i' 'a' '\0' ? ...
    prato   array
             'H' 'a' 'm' 'b' 'u' 'r' 'g' 'u' 'e' 'r' '\0' ? ...
    quant   array
             1
    status   array
             'e' 'm' ' ' 'p' 'r' 'e' 'p' 'a' 'r' 'o'
    prox   pointer to Pedido
             ?

```
17    if (novoPedido == NULL) {
18        printf("Erro ao alocar memória!\n");
19        return;
20    }
21    novoPedido->num = num;
22    strcpy(novoPedido->nomeCliente, nomeCliente);
23    strcpy(novoPedido->prato, prato);
24    novoPedido->quant = quant;
25    strcpy(novoPedido->status, status);
26    novoPedido->prox = *inicio;
27    *inicio = novoPedido;
28
29    printf("Pedido #%d inserido!\n", num);
30 }
31 void obterPedido(struct Pedido *inicio, int num) {
32     struct Pedido *ptr = inicio;
33     while (ptr != NULL) {
```

Edit this code

→ line that just executed
➡ next line to execute

<< First | < Prev | Next > | Last >>
Step 27 of 76

Move and hide objects

Stack   Heap

main
    inicio   pointer to Pedido

inserirPedido
    inicio   pointer to Pedido*
    num   int
                102
    nomeCliente pointer to char
    prato   pointer to char
    quant   int
                1
    status   pointer to char
    novoPedido pointer to Pedido

array
struct Pedido
    num   int
             101
    nomeCliente array
             'J' 'o' '\xC3' '\xA3' 'o' '\0' ? ...
    prato   array
             'P' 'i' 'z' 'z' 'a' '\0' ? ...
    quant   int
             2
    status   array
             'p' 'e' 'n' 'd' 'e' 'n' 't' 'e' '\0' ?
    prox   pointer to Pedido
             NULL (0x0)

array
'M' 'a' 'r' 'i' 'a' '\0'
(this is in read-only storage, not the heap)

array
'H' 'a' 'm' 'b' 'u' 'r' 'g' 'u' 'e' 'r' '\0'
(this is in read-only storage, not the heap)

array
'e' 'm' ' ' 'p' 'r' 'e' 'p' 'a' 'r' 'o' '\0'
(this is in read-only storage, not the heap)

array
struct Pedido
    num   int
             102
    nomeCliente array
             'M' 'a' 'r' 'i' 'a' '\0' ? ...
    prato   array
             'H' 'a' 'm' 'b' 'u' 'r' 'g' 'u' 'e' 'r' '\0' ? ...
    quant   array
             1
    status   array
             'e' 'm' ' ' 'p' 'r' 'e' 'p' 'a' 'r' 'o'
    prox   pointer to Pedido
```

Python Tutor: Visualize Code and Get AI Help for Python, JavaScript, C, C++, and Java

C (C17 + GNU extensions)
known limitations

```
75    struct Pedido *atual = *inicio;
76    struct Pedido *prox;
77    while (atual != NULL) {
78        prox = atual->prox;
79        free(atual);
80        atual = prox;
81    }
82    *inicio = NULL;
83    printf("Lista de pedidos liberada!\n");
84 }
85 int main() {
86    struct Pedido *inicio = criarLista();
87    inserirPedido(&inicio, 101, "João", "Pizza", 2,
88    inserirPedido(&inicio, 102, "Maria", "Hamburguer
89    obterPedido(inicio, 102);
90    atualizarStatus(inicio, 101, "entregue");
91    deletarPedido(&inicio, 102);
92    liberarLista(&inicio);
93    return 0;
94 }
```

Edit this code

⟶ line that just executed
➡ next line to execute

<< First   < Prev   Next >   Last >>
Step 30 of 76

follow our YouTube, TikTok, Instagram for weekly tutorials
Move and hide objects

Print output (drag lower right corner to resize)
Pedido #101 inserido!
Pedido #102 inserido!

Stack   Heap

Obter Pedido

Python Tutor: Visualize Code and Get AI Help for Python, JavaScript, C, C++, and Java

C (C17 + GNU extensions)
known limitations

```
21    novoPedido->num = num;
22    strcpy(novoPedido->nomeCliente, nomeCliente);
23    strcpy(novoPedido->prato, prato);
24    novoPedido->quant = quant;
25    strcpy(novoPedido->status, status);
26    novoPedido->prox = *inicio;
27    *inicio = novoPedido;
28
29    printf("Pedido #%d inserido!\n", num);
30 }
31 void obterPedido(struct Pedido *inicio, int num) {
32    struct Pedido *ptr = inicio;
33    while (ptr != NULL) {
34        if (ptr->num == num) {
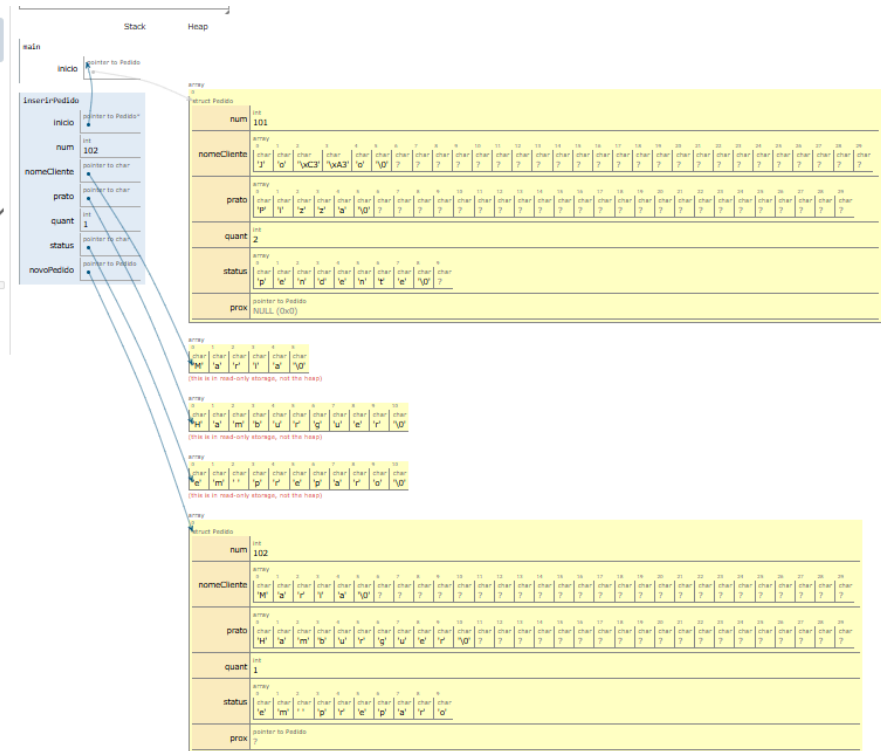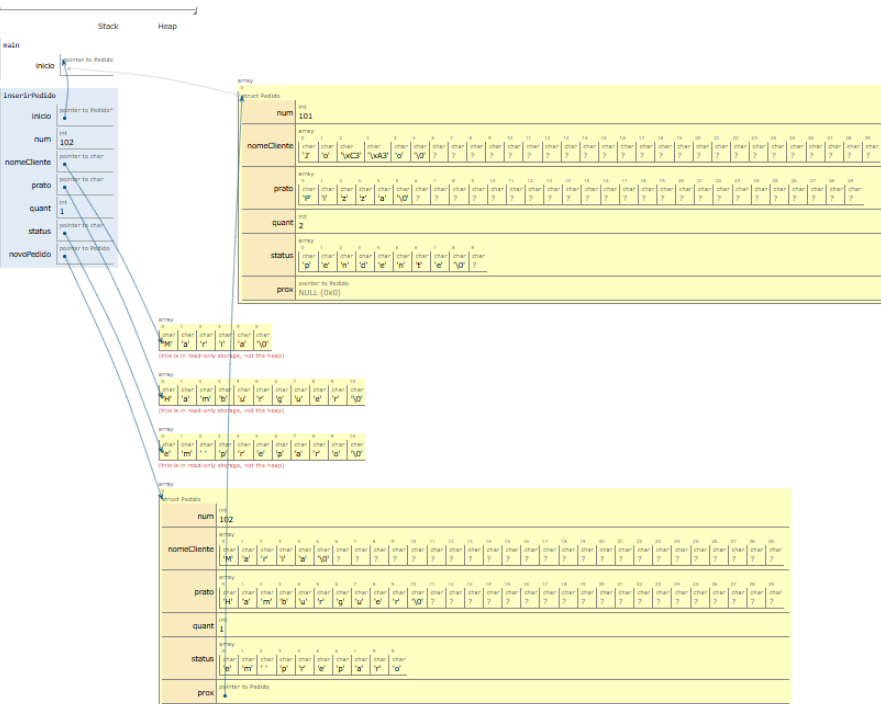35            printf("Pedido #%d encontrado!\nCliente:
36                ptr->num, ptr->nomeCliente, ptr->;
37            return;
38        }
39        ptr = ptr->prox;
```

Edit this code

⟶ line that just executed
➡ next line to execute

<< First   < Prev   Next >   Last >>
Step 31 of 76

follow our YouTube, TikTok, Instagram for weekly tutorials
Move and hide objects

Print output (drag lower right corner to resize)
Pedido #101 inserido!
Pedido #102 inserido!

Stack   Heap

C (C17 + GNU extensions)
known limitations

```
32      struct Pedido *ptr = inicio;
33      while (ptr != NULL) {
34          if (ptr->num == num) {
35              printf("Pedido #%d encontrado!\nCliente:
36                  ptr->num, ptr->nomeCliente, ptr->
37              return;
38          }
39          ptr = ptr->prox;
40      }
41      printf("Pedido #%d não encontrado!\n", num);
42  }
43  void atualizarStatus(struct Pedido *inicio, int num,
44      struct Pedido *ptr = inicio;
45      while (ptr != NULL) {
46          if (ptr->num == num) {
47              strcpy(ptr->status, novoStatus);
48              printf("Status do Pedido #%d atualizado
49              return;
50          }
```

Edit this code

→ line that just executed
➡ next line to execute

<< First   < Prev   Next >   Last >>
Step 40 of 76

follow our YouTube, TikTok, Instagram for weekly tutorials

Move and hide objects

Print output (drag lower right corner to resize)
```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
```

Stack   Heap

Atualizar Status

C (C17 + GNU extensions)
known limitations

```
32      struct Pedido *ptr = inicio;
33      while (ptr != NULL) {
34          if (ptr->num == num) {
35              printf("Pedido #%d encontrado!\nCliente:
36                  ptr->num, ptr->nomeCliente, ptr->
37              return;
38          }
39          ptr = ptr->prox;
40      }
41      printf("Pedido #%d não encontrado!\n", num);
42  }
43  void atualizarStatus(struct Pedido *inicio, int num,
44      struct Pedido *ptr = inicio;
45      while (ptr != NULL) {
46          if (ptr->num == num) {
47              strcpy(ptr->status, novoStatus);
48              printf("Status do Pedido #%d atualizado
49              return;
50          }
51          ptr = ptr->prox;
```

Edit this code

→ line that just executed
➡ next line to execute

<< First   < Prev   Next >   Last >>
Step 42 of 76

follow our YouTube, TikTok, Instagram for weekly tutorials

Move and hide objects

Print output (drag lower right corner to resize)
```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
```

Stack   Heap

Print output (drag lower right corner to resize)

```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
```

```c
    struct Pedido *ptr = inicio;
    while (ptr != NULL) {
        if (ptr->num == num) {
            printf("Pedido #%d encontrado!\nCliente:
                ptr->num, ptr->nomeCliente, ptr->
            return;
        }
        ptr = ptr->prox;
    }
    printf("Pedido #%d não encontrado!\n", num);
}
void atualizarStatus(struct Pedido *inicio, int num,
    struct Pedido *ptr = inicio;
    while (ptr != NULL) {
        if (ptr->num == num) {
            strcpy(ptr->status, novoStatus);
            printf("Status do Pedido #%d atualizado
            return;
        }
        ptr = ptr->prox;
```

Edit this code

⇒ line that just executed
➡ next line to execute

<< First   < Prev   Next >   Last >>

Step 43 of 76

follow our **YouTube**, **TikTok**, **Instagram** for weekly tutorials

Move and hide objects

Stack   **Heap**

main
  inicio — pointer to Pedido

atualizarStatus
  inicio — pointer to Pedido
  num — 101
  novoStatus — pointer to char
  ptr — pointer to Pedido

struct Pedido
  num: 101
  nomeCliente: 'J' 'o' '\xC3' '\xA3' 'o' '\0' ? ...
  prato: 'P' 'i' 'z' 'a' '\0' ? ...
  quant: 2
  status: 'p' 'e' 'n' 'd' 'e' 'n' 't' 'e' '\0' ?
  prox: NULL (0x0)

struct Pedido
  num: 102
  nomeCliente: 'M' 'a' 'r' 'i' 'a' '\0' ? ...
  prato: 'H' 'a' 'm' 'b' 'u' 'r' 'g' 'u' 'e' 'r' '\0' ? ...
  quant: 1
  status: 'e' 'm' ' ' 'p' 'r' 'e' 'p' 'a' 'r' 'o'
  prox: pointer to Pedido

array
  'e' 'n' 't' 'r' 'e' 'g' 'u' 'e' '\0'
  (this is in read-only storage, not the heap)

Print output (drag lower right corner to resize)

```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
```

```c
    struct Pedido *ptr = inicio;
    while (ptr != NULL) {
        if (ptr->num == num) {
            printf("Pedido #%d encontrado!\nCliente:
                ptr->num, ptr->nomeCliente, ptr->
            return;
        }
        ptr = ptr->prox;
    }
    printf("Pedido #%d não encontrado!\n", num);
}
void atualizarStatus(struct Pedido *inicio, int num,
    struct Pedido *ptr = inicio;
    while (ptr != NULL) {
        if (ptr->num == num) {
            strcpy(ptr->status, novoStatus);
            printf("Status do Pedido #%d atualizado
            return;
        }
        ptr = ptr->prox;
```

Edit this code

⇒ line that just executed
➡ next line to execute

<< First   < Prev   Next >   Last >>

Step 46 of 76

follow our **YouTube**, **TikTok**, **Instagram** for weekly tutorials

Move and hide objects

Stack   **Heap**

main
  inicio — pointer to Pedido

atualizarStatus
  inicio — pointer to Pedido
  num — 101
  novoStatus — pointer to char
  ptr — pointer to Pedido

struct Pedido
  num: 101
  nomeCliente: 'J' 'o' '\xC3' '\xA3' 'o' '\0' ? ...
  prato: 'P' 'i' 'z' 'a' '\0' ? ...
  quant: 2
  status: 'p' 'e' 'n' 'd' 'e' 'n' 't' 'e' '\0' ?
  prox: NULL (0x0)

struct Pedido
  num: 102
  nomeCliente: 'M' 'a' 'r' 'i' 'a' '\0' ? ...
  prato: 'H' 'a' 'm' 'b' 'u' 'r' 'g' 'u' 'e' 'r' '\0' ? ...
  quant: 1
  status: 'e' 'm' ' ' 'p' 'r' 'e' 'p' 'a' 'r' 'o'
  prox: pointer to Pedido

array
  'e' 'n' 't' 'r' 'e' 'g' 'u' 'e' '\0'
  (this is in read-only storage, not the heap)

```
32    struct Pedido *ptr = inicio;
33    while (ptr != NULL) {
34        if (ptr->num == num) {
35            printf("Pedido #%d encontrado!\nCliente:
36                ptr->num, ptr->nomeCliente, ptr->:
37            return;
38        }
39        ptr = ptr->prox;
40    }
41    printf("Pedido #%d não encontrado!\n", num);
42 }
43 void atualizarStatus(struct Pedido *inicio, int num,
44    struct Pedido *ptr = inicio;
45    while (ptr != NULL) {
46        if (ptr->num == num) {
47            strcpy(ptr->status, novoStatus);
48            printf("Status do Pedido #%d atualizado
49            return;
50        }
51        ptr = ptr->prox;
```

Edit this code

⇒ line that just executed
➡ next line to execute

<< First | < Prev | Next > | Last >>

Step 47 of 76

Print output (drag lower right corner to resize)
```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
```

Stack   Heap

main
  inicio   pointer to Pedidos

atualizarStatus
  inicio   pointer to Pedidos
  num   int 101
  novoStatus pointer to char
  ptr   pointer to Pedidos

struct Pedido
  num   int 101
  nomeCliente 'J' 'o' '\xC3' '\xA3' 'o' '\0' ...
  prato   'P' 'i' 'z' 'z' 'a' '\0' ...
  quant   int 2
  status   'p' 'e' 'n' 'd' 'e' 'n' 't' 'e' '\0'
  prox   pointer to Pedido NULL (0x0)

struct Pedido
  num   int 102
  nomeCliente 'M' 'a' 'r' 'i' 'a' '\0' ...
  prato   'H' 'a' 'm' 'b' 'u' 'r' 'g' 'u' 'e' 'r' '\0' ...
  quant   int 1
  status   'e' 'm' ' ' 'p' 'r' 'e' 'p' 'a' 'r' 'o'
  prox   pointer to Pedido

'e' 'n' 't' 'r' 'e' 'g' 'u' 'e' '\0'
(this is in read-only storage, not the heap)

```
32    struct Pedido *ptr = inicio;
33    while (ptr != NULL) {
34        if (ptr->num == num) {
35            printf("Pedido #%d encontrado!\nCliente:
36                ptr->num, ptr->nomeCliente, ptr->:
37            return;
38        }
39        ptr = ptr->prox;
40    }
41    printf("Pedido #%d não encontrado!\n", num);
42 }
43 void atualizarStatus(struct Pedido *inicio, int num,
44    struct Pedido *ptr = inicio;
45    while (ptr != NULL) {
46        if (ptr->num == num) {
47            strcpy(ptr->status, novoStatus);
48            printf("Status do Pedido #%d atualizado
49            return;
50        }
51        ptr = ptr->prox;
```

Edit this code

⇒ line that just executed
➡ next line to execute

<< First | < Prev | Next > | Last >>

Step 50 of 76

Print output (drag lower right corner to resize)
```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
```

Stack   Heap

main
  inicio   pointer to Pedidos

atualizarStatus
  inicio   pointer to Pedidos
  num   int 101
  novoStatus pointer to char
  ptr   pointer to Pedidos

struct Pedido
  num   int 101
  nomeCliente 'J' 'o' '\xC3' '\xA3' 'o' '\0' ...
  prato   'P' 'i' 'z' 'z' 'a' '\0' ...
  quant   int 2
  status   'e' 'n' 't' 'r' 'e' 'g' 'u' 'e' '\0'
  prox   pointer to Pedido NULL (0x0)

struct Pedido
  num   int 102
  nomeCliente 'M' 'a' 'r' 'i' 'a' '\0' ...
  prato   'H' 'a' 'm' 'b' 'u' 'r' 'g' 'u' 'e' 'r' '\0' ...
  quant   int 1
  status   'e' 'm' ' ' 'p' 'r' 'e' 'p' 'a' 'r' 'o'
  prox   pointer to Pedido

'e' 'n' 't' 'r' 'e' 'g' 'u' 'e' '\0'
(this is in read-only storage, not the heap)

Deletar Pedido

C (C17 + GNU extensions)
known limitations

```
44    struct Pedido *ptr = inicio;
45    while (ptr != NULL) {
46        if (ptr->num == num) {
47            strcpy(ptr->status, novoStatus);
48            printf("Status do Pedido #%d atualizado (
49            return;
50        }
51        ptr = ptr->prox;
52    }
53    printf("Pedido #%d não encontrado!\n", num);
54 }
55 void deletarPedido(struct Pedido **inicio, int num) ·
56    struct Pedido *atual = *inicio;
57    struct Pedido *anterior = NULL;
58    while (atual != NULL && atual->num != num) {
59        anterior = atual;
60        atual = atual->prox;
61    }
62    if (atual == NULL) {
63        printf("Pedido #%d não encontrado!\n", num);
```

Edit this code

→ line that just executed
➡ next line to execute

[<< First] [< Prev] [Next >] [Last >>]
Step 54 of 76

follow our YouTube, TikTok, Instagram for weekly tutorials
Move and hide objects

Print output (drag lower right corner to resize)
```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
Status do Pedido #101 atualizado para: entregue
```

Stack   Heap

C (C17 + GNU extensions)
known limitations

```
44    struct Pedido *ptr = inicio;
45    while (ptr != NULL) {
46        if (ptr->num == num) {
47            strcpy(ptr->status, novoStatus);
48            printf("Status do Pedido #%d atualizado (
49            return;
50        }
51        ptr = ptr->prox;
52    }
53    printf("Pedido #%d não encontrado!\n", num);
54 }
55 void deletarPedido(struct Pedido **inicio, int num) ·
56    struct Pedido *atual = *inicio;
57    struct Pedido *anterior = NULL;
58    while (atual != NULL && atual->num != num) {
59        anterior = atual;
60        atual = atual->prox;
61    }
62    if (atual == NULL) {
63        printf("Pedido #%d não encontrado!\n", num);
```

Edit this code

→ line that just executed
➡ next line to execute

[<< First] [< Prev] [Next >] [Last >>]
Step 58 of 76

follow our YouTube, TikTok, Instagram for weekly tutorials
Move and hide objects

Print output (drag lower right corner to resize)
```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
Status do Pedido #101 atualizado para: entregue
```

Stack   Heap

C (C17 + GNU extensions)
known limitations

```
55 void deletarPedido(struct Pedido **inicio, int num) ·
56    struct Pedido *atual = *inicio;
57    struct Pedido *anterior = NULL;
58    while (atual != NULL && atual->num != num) {
59        anterior = atual;
60        atual = atual->prox;
61    }
62    if (atual == NULL) {
63        printf("Pedido #%d não encontrado!\n", num);
64        return;
65    }
66    if (anterior == NULL) {
67        *inicio = atual->prox;
68    } else {
69        anterior->prox = atual->prox;
70    }
71    free(atual);
72    printf("Pedido #%d deletado com sucesso!\n", num)
73 }
74 void liberarLista(struct Pedido **inicio) {
```

Edit this code

→ line that just executed
➡ next line to execute

[<< First] [< Prev] [Next >] [Last >>]
Step 61 of 76

follow our YouTube, TikTok, Instagram for weekly tutorials
Move and hide objects

Print output (drag lower right corner to resize)
```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
Status do Pedido #101 atualizado para: entregue
```

Stack   Heap

C (C17 + GNU extensions)
known limitations

```
55  void deletarPedido(struct Pedido **inicio, int num) {
56      struct Pedido *atual = *inicio;
57      struct Pedido *anterior = NULL;
58      while (atual != NULL && atual->num != num) {
59          anterior = atual;
60          atual = atual->prox;
61      }
62      if (atual == NULL) {
63          printf("Pedido #%d não encontrado!\n", num);
64          return;
65      }
66      if (anterior == NULL) {
67          *inicio = atual->prox;
68      } else {
69          anterior->prox = atual->prox;
70      }
71      free(atual);
72      printf("Pedido #%d deletado com sucesso!\n", num);
73  }
74  void liberarLista(struct Pedido **inicio) {
```

Edit this code

→ line that just executed
➡ next line to execute

Step 63 of 76

<< First   < Prev   Next >   Last >>

Move and hide objects

Print output (drag lower right corner to resize)
```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
Status do Pedido #101 atualizado para: entregue
Pedido #102 deletado com sucesso!
```

Stack   Heap

Liberar Lista

C (C17 + GNU extensions)
known limitations

```
63      printf("Pedido #%d não encontrado!\n", num);
64      return;
65  }
66  if (anterior == NULL) {
67      *inicio = atual->prox;
68  } else {
69      anterior->prox = atual->prox;
70  }
71  free(atual);
72  printf("Pedido #%d deletado com sucesso!\n", num);
73  }
74  void liberarLista(struct Pedido **inicio) {
75      struct Pedido *atual = *inicio;
76      struct Pedido *prox;
77      while (atual != NULL) {
78          prox = atual->prox;
79          free(atual);
80          atual = prox;
81      }
82      *inicio = NULL;
```

Edit this code

→ line that just executed
➡ next line to execute

Step 65 of 76

<< First   < Prev   Next >   Last >>

Move and hide objects

Print output (drag lower right corner to resize)
```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
Status do Pedido #101 atualizado para: entregue
Pedido #102 deletado com sucesso!
```

Stack   Heap

C (C17 + GNU extensions)
known limitations

```
63      printf("Pedido #%d não encontrado!\n", num);
64      return;
65  }
66  if (anterior == NULL) {
67      *inicio = atual->prox;
68  } else {
69      anterior->prox = atual->prox;
70  }
71  free(atual);
72  printf("Pedido #%d deletado com sucesso!\n", num);
73  }
74  void liberarLista(struct Pedido **inicio) {
75      struct Pedido *atual = *inicio;
76      struct Pedido *prox;
77      while (atual != NULL) {
78          prox = atual->prox;
79          free(atual);
80          atual = prox;
81      }
82      *inicio = NULL;
```

Edit this code

→ line that just executed
➡ next line to execute

Step 69 of 76

<< First   < Prev   Next >   Last >>

Move and hide objects

Print output (drag lower right corner to resize)
```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
Status do Pedido #101 atualizado para: entregue
Pedido #102 deletado com sucesso!
```

Stack   Heap

C (C17 + GNU extensions)
known limitations

```c
64        return;
65    }
66    if (anterior == NULL) {
67        *inicio = atual->prox;
68    } else {
69        anterior->prox = atual->prox;
70    }
71    free(atual);
72    printf("Pedido #%d deletado com sucesso!\n", num
73 }
74 void liberarLista(struct Pedido **inicio) {
75    struct Pedido *atual = *inicio;
76    struct Pedido *prox;
77    while (atual != NULL) {
→ 78        prox = atual->prox;
→ 79        free(atual);
80        atual = prox;
81    }
82    *inicio = NULL;
```

Edit this code

→ line that just executed
→ next line to execute

<< First | < Prev | Next > | Last >>
Step 69 of 76

follow our **YouTube**, **TikTok**, **Instagram** for weekly tutorials
Move and hide objects

Print output (drag lower right corner to resize)
```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
Status do Pedido #101 atualizado para: entregue
Pedido #102 deletado com sucesso!
```

Stack   Heap

main

inicio [pointer to Pedido]

array
struct Pedido

liberarLista

inicio [pointer to Pedido*]

atual [pointer to Pedido]

prox [pointer to Pedido] NULL (0x0)

num [int] 101

nomeCliente [array] 'J' 'o' '\xC3' '\xA3' 'o' '\0' ? ? ? ...

prato [array] 'P' 'i' 'z' 'z' 'a' '\0' ? ? ...

quant [int] 2

status [array] 'e' 'n' 't' 'r' 'e' 'g' 'u' 'e' '\0' ?

prox [pointer to Pedido] NULL (0x0)

C (C17 + GNU extensions)
known limitations

```c
64        return;
65    }
66    if (anterior == NULL) {
67        *inicio = atual->prox;
68    } else {
69        anterior->prox = atual->prox;
70    }
71    free(atual);
72    printf("Pedido #%d deletado com sucesso!\n", num
73 }
74 void liberarLista(struct Pedido **inicio) {
75    struct Pedido *atual = *inicio;
76    struct Pedido *prox;
77    while (atual != NULL) {
78        prox = atual->prox;
→ 79        free(atual);
→ 80        atual = prox;
81    }
82    *inicio = NULL;
```

Edit this code

→ line that just executed
→ next line to execute

<< First | < Prev | Next > | Last >>
Step 70 of 76

Print output (drag lower right corner to resize)
```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
Status do Pedido #101 atualizado para: entregue
Pedido #102 deletado com sucesso!
```

Stack   Heap

main

inicio [pointer to Pedido]

liberarLista

inicio [pointer to Pedido*]

atual [pointer to Pedido]

prox [pointer to Pedido] NULL (0x0)

Note: 👀 means a pointer points to memory that is either unallocated or misaligned with data boundaries. 👀 locations are *approximate* and may not match the pointer's real address. Select 'byte-level view of data' below to see more details:

C/C++ details: [none [default view] ▾]

Python Tutor: Visualize Code and Get AI Help for Python, JavaScript, C, C++, and Java

```
63        printf("Pedido #%d não encontrado!\n", num);
64        return;
65    }
66    if (anterior == NULL) {
67        *inicio = atual->prox;
68    } else {
69        anterior->prox = atual->prox;
70    }
71    free(atual);
72    printf("Pedido #%d deletado com sucesso!\n", num);
73 }
74 void liberarLista(struct Pedido **inicio) {
75    struct Pedido *atual = *inicio;
76    struct Pedido *prox;
77    while (atual != NULL) {
78        prox = atual->prox;
79        free(atual);
80        atual = prox;
81    }
82    *inicio = NULL;
83    printf("Lista de pedidos liberada!\n");
```

Edit this code

→ line that just executed
→ next line to execute

<< First   < Prev   Next >   Last >>

Step 73 of 76

Print output (drag lower right corner to resize)

```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
Status do Pedido #101 atualizado para: entregue
Pedido #102 deletado com sucesso!
```

Stack   Heap

```
main
       pointer to Pedido
inicio NULL (0x0)

liberarLista
       pointer to Pedido*
inicio

       pointer to Pedido
atual  NULL (0x0)

       pointer to Pedido
prox   NULL (0x0)
```

C/C++ details: none [default view]

Print output (drag lower right corner to resize)

```
Pedido #101 inserido!
Pedido #102 inserido!
Pedido #102 encontrado!
Cliente: Maria, Prato: Hamburguer, Quantidade: 1, Status: em preparo
Status do Pedido #101 atualizado para: entregue
Pedido #102 deletado com sucesso!
Lista de pedidos liberada!
```