Aula 26 – Estruturas (parte 2)

Norton T. Roman & Luciano A. Digiampietri

Estruturas e Funções

- As estruturas funcionam de maneira similar a qualquer outro tipo de dados

Estruturas e Funções

- As estruturas funcionam de maneira similar a qualquer outro tipo de dados
  - Podemos criar variáveis do seu tipo

Estruturas e Funções

- As estruturas funcionam de maneira similar a qualquer outro tipo de dados
  - Podemos criar variáveis do seu tipo
  - Podemos alocar memória dinamicamente para armazenar dados do "tipo" estrutura

Estruturas e Funções

- As estruturas funcionam de maneira similar a qualquer outro tipo de dados
  - Podemos criar variáveis do seu tipo
  - Podemos alocar memória dinamicamente para armazenar dados do "tipo" estrutura
  - Os parâmetros de funções podem ser do tipo estrutura

Estruturas e Funções

- As estruturas funcionam de maneira similar a qualquer outro tipo de dados
  - Podemos criar variáveis do seu tipo
  - Podemos alocar memória dinamicamente para armazenar dados do "tipo" estrutura
  - Os parâmetros de funções podem ser do tipo estrutura
  - O retorno de funções pode ser do tipo estrutura

Estruturas e Funções

```c
#include <stdio.h>

typedef struct auxCasa {
  float lateral;
  float cquarto;
} casa;
```

```c
void areaCasa(casa pCasa) {
   float areaq;
   float areas;
   float areat;
   if (!(pCasa.lateral>=0 && pCasa.cquarto>=0))
      printf("Erro: parametro < 0\n");
   else {
      printf("Programa para calculo da area
                                        da casa\n");
      areas = pCasa.lateral*pCasa.lateral;
      printf("A area da sala e %f\n", areas);
      areaq = pCasa.cquarto*(pCasa.lateral/2);
      printf("A area do quarto e %f\n", areaq);
      printf("A area do banheiro e %f\n", areaq);
      areat = areas + 2*areaq;
      printf("A area total e %f\n", areat);
   }
}
```

Estruturas e Funções

```c
#include <stdio.h>

typedef struct auxCasa {
  float lateral;
  float cquarto;
} casa;
```

```c
void areaCasa(casa pCasa) {
   float areaq;
   float areas;
   float areat;
   if (!(pCasa.lateral>=0 && pCasa.cquarto>=0))
      printf("Erro: parametro < 0\n");
   else {
      printf("Programa para calculo da area
                                    da casa\n");
      areas = pCasa.lateral*pCasa.lateral;
      printf("A area da sala e %f\n", areas);
      areaq = pCasa.cquarto*(pCasa.lateral/2);
      printf("A area do quarto e %f\n", areaq);
      printf("A area do banheiro e %f\n", areaq);
      areat = areas + 2*areaq;
      printf("A area total e %f\n", areat);
   }
}
```

Estruturas e Funções

```c
#include <stdio.h>

typedef struct auxCasa {
  float lateral;
  float cquarto;
} casa;
```

```c
void areaCasa(casa pCasa) {
   float areaq;
   float areas;
   float areat;
   if (!(pCasa.lateral>=0 && pCasa.cquarto>=0))
      printf("Erro: parametro < 0\n");
   else {
      printf("Programa para calculo da area
                                        da casa\n");
      areas = pCasa.lateral*pCasa.lateral;
      printf("A area da sala e %f\n", areas);
      areaq = pCasa.cquarto*(pCasa.lateral/2);
      printf("A area do quarto e %f\n", areaq);
      printf("A area do banheiro e %f\n", areaq);
      areat = areas + 2*areaq;
      printf("A area total e %f\n", areat);
   }
}
```

Estruturas e Funções

```c
#include <stdio.h>

typedef struct auxCasa {
  float lateral;
  float cquarto;
} casa;
```

```c
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  areaCasa(c1);
  return 0;
}
```

```c
void areaCasa(casa pCasa) {
   float areaq;
   float areas;
   float areat;
   if (!(pCasa.lateral>=0 && pCasa.cquarto>=0))
      printf("Erro: parametro < 0\n");
   else {
      printf("Programa para calculo da area
                                        da casa\n");
      areas = pCasa.lateral*pCasa.lateral;
      printf("A area da sala e %f\n", areas);
      areaq = pCasa.cquarto*(pCasa.lateral/2);
      printf("A area do quarto e %f\n", areaq);
      printf("A area do banheiro e %f\n", areaq);
      areat = areas + 2*areaq;
      printf("A area total e %f\n", areat);
   }
}
```

Estruturas e Funções

```c
#include <stdio.h>

typedef struct auxCasa {
  float lateral;
  float cquarto;
} casa;
```

```c
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  areaCasa(c1);
  return 0;
}
```

```c
void areaCasa(casa pCasa) {
   float areaq;
   float areas;
   float areat;
   if (!(pCasa.lateral>=0 && pCasa.cquarto>=0))
      printf("Erro: parametro < 0\n");
   else {
      printf("Programa para calculo da area
                                        da casa\n");
      areas = pCasa.lateral*pCasa.lateral;
      printf("A area da sala e %f\n", areas);
      areaq = pCasa.cquarto*(pCasa.lateral/2);
      printf("A area do quarto e %f\n", areaq);
      printf("A area do banheiro e %f\n", areaq);
      areat = areas + 2*areaq;
      printf("A area total e %f\n", areat);
   }
}
```

Saída:

```
Programa para calculo da area da casa
A area da sala e 121.000000
A area do quarto e 82.500000
A area do banheiro e 82.500000
A area total e 286.000000
```

Estruturas e Funções

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  areaCasa(c1);
  return 0;
}
```

- Podemos criar uma função para facilitar a inicialização dos dados de uma estrutura

Estruturas e Funções

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  areaCasa(c1);
  return 0;
}
```

- Podemos criar uma função para facilitar a inicialização dos dados de uma estrutura
- Ela será responsável por criar e inicializar uma casa e retorná-la

Estruturas e Funções

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  areaCasa(c1);
  return 0;
}
```

```
casa iniciaCasa(float lateral, float cquarto) {
  casa resp;
  resp.lateral = lateral;
  resp.cquarto = cquarto;
  return resp;
}
```

- Podemos criar uma função para facilitar a inicialização dos dados de uma estrutura
- Ela será responsável por criar e inicializar uma casa e retorná-la

Estruturas e Funções

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  areaCasa(c1);
  return 0;
}
```

```
casa iniciaCasa(float lateral, float cquarto) {
  casa resp;
  resp.lateral = lateral;
  resp.cquarto = cquarto;
  return resp;
}
```

- Podemos criar uma função para facilitar a inicialização dos dados de uma estrutura
- Ela será responsável por criar e inicializar uma casa e retorná-la

Estruturas e Funções

```
int main() {
  casa c1 = iniciaCasa(11,15);


  areaCasa(c1);
  return 0;
}
```

```
casa iniciaCasa(float lateral, float cquarto) {
  casa resp;
  resp.lateral = lateral;
  resp.cquarto = cquarto;
  return resp;
}
```

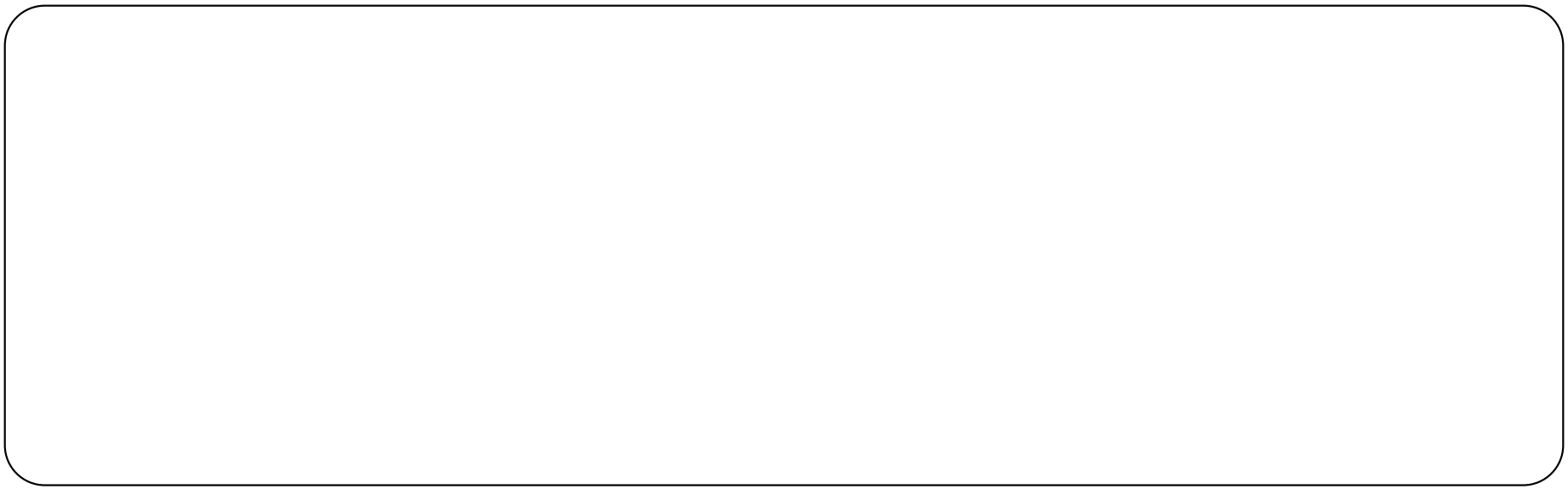
- Podemos criar uma função para facilitar a inicialização dos dados de uma estrutura
- Ela será responsável por criar e inicializar uma casa e retorná-la

Estruturas e Memória

```
int main() {
  casa c1 = iniciaCasa(11,15);

  return 0;
}
```

```
casa iniciaCasa(float lateral, float cquarto) {
  casa resp;
  resp.lateral = lateral;
  resp.cquarto = cquarto;
  return resp;
}
```

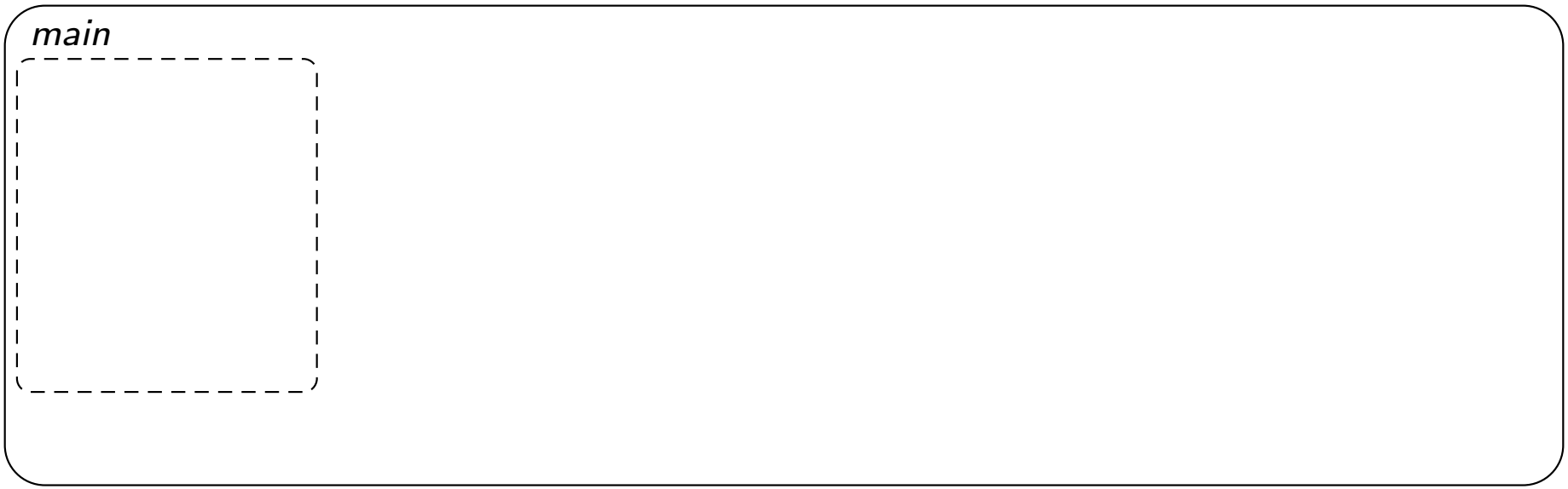


Estruturas e Memória

```
int main() {
  casa c1 = iniciaCasa(11,15);

  return 0;
}
```

```
casa iniciaCasa(float lateral, float cquarto) {
  casa resp;
  resp.lateral = lateral;
  resp.cquarto = cquarto;
  return resp;
}
```

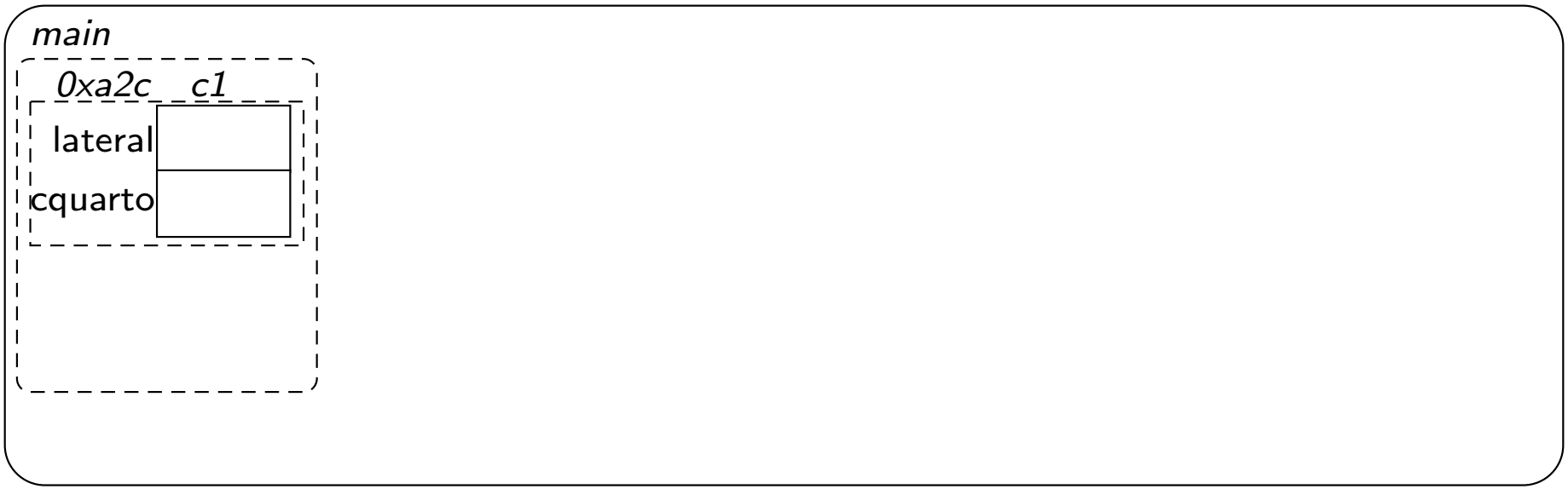


Estruturas e Memória

```
int main() {
  casa c1 = iniciaCasa(11,15);

  return 0;
}
```

```
casa iniciaCasa(float lateral, float cquarto) {
  casa resp;
  resp.lateral = lateral;
  resp.cquarto = cquarto;
  return resp;
}
```

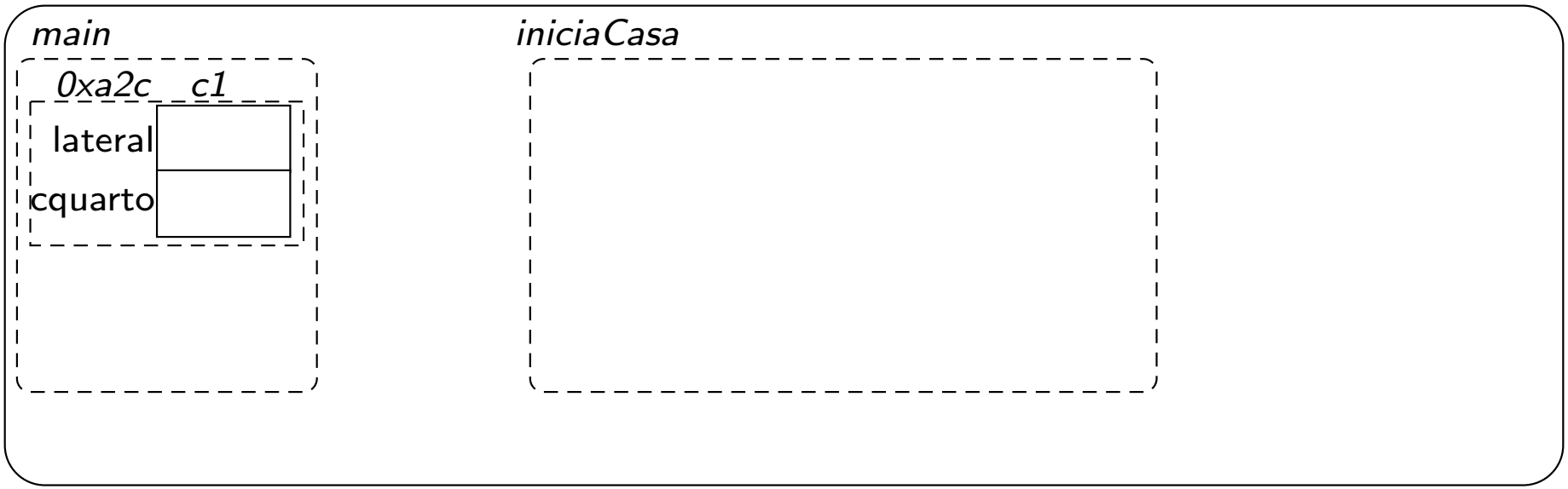


Estruturas e Memória

```
int main() {
  casa c1 = iniciaCasa(11,15);

  return 0;
}
```

```
casa iniciaCasa(float lateral, float cquarto) {
  casa resp;
  resp.lateral = lateral;
  resp.cquarto = cquarto;
  return resp;
}
```

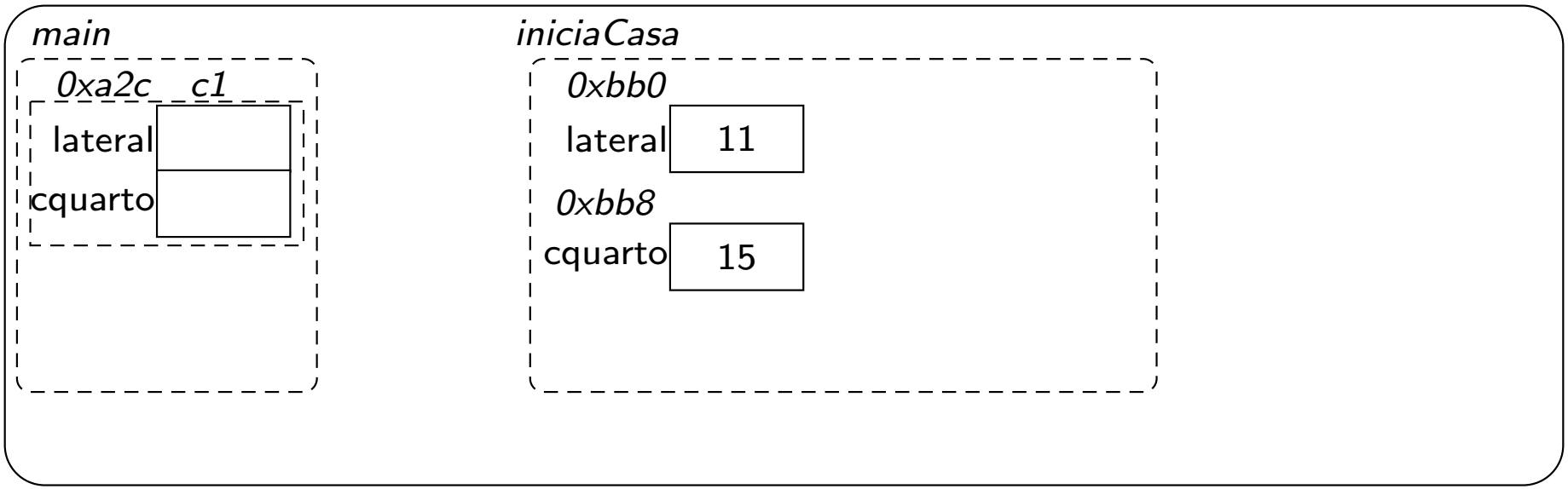


Estruturas e Memória

```
int main() {
  casa c1 = iniciaCasa(11,15);

  return 0;
}
```

```
casa iniciaCasa(float lateral, float cquarto) {
  casa resp;
  resp.lateral = lateral;
  resp.cquarto = cquarto;
  return resp;
}
```

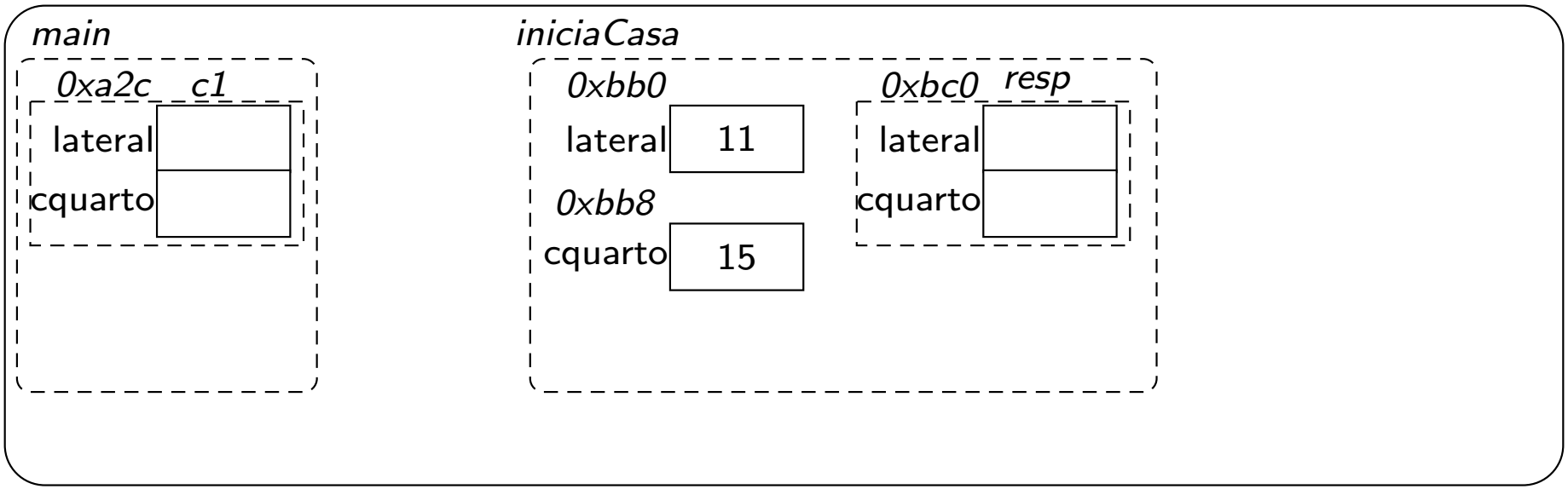


Estruturas e Memória

```
int main() {
  casa c1 = iniciaCasa(11,15);

  return 0;
}
```

```
casa iniciaCasa(float lateral, float cquarto) {
  casa resp;
  resp.lateral = lateral;
  resp.cquarto = cquarto;
  return resp;
}
```

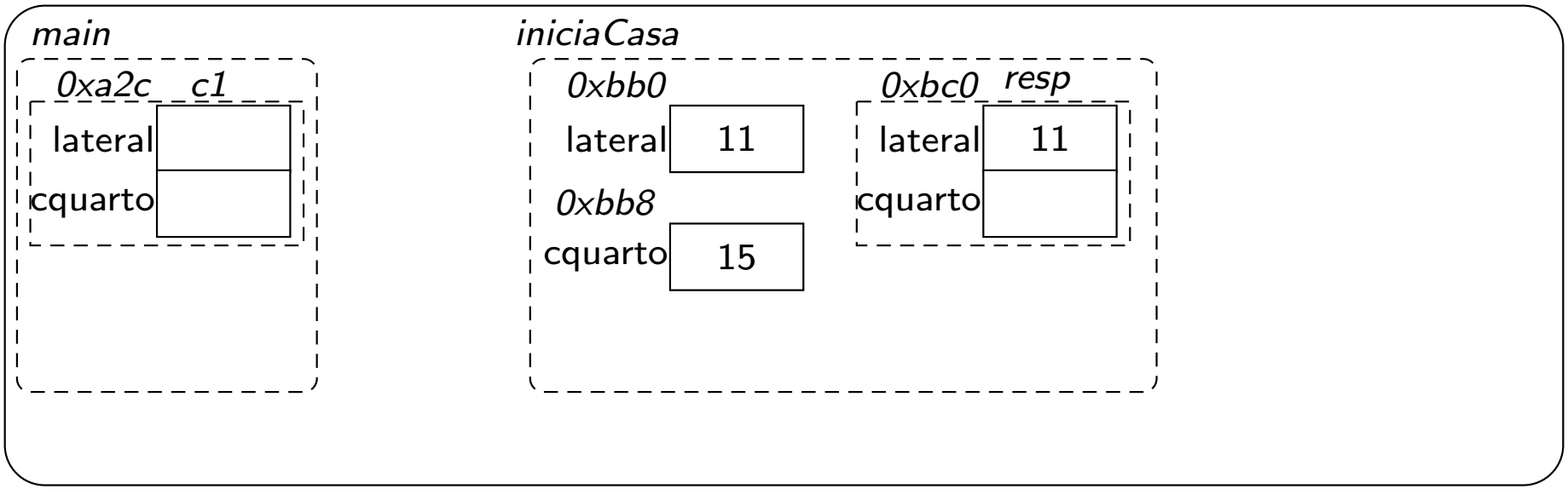


Estruturas e Memória

```
int main() {
  casa c1 = iniciaCasa(11,15);

  return 0;
}
```

```
casa iniciaCasa(float lateral, float cquarto) {
  casa resp;
  resp.lateral = lateral;
  resp.cquarto = cquarto;
  return resp;
}
```

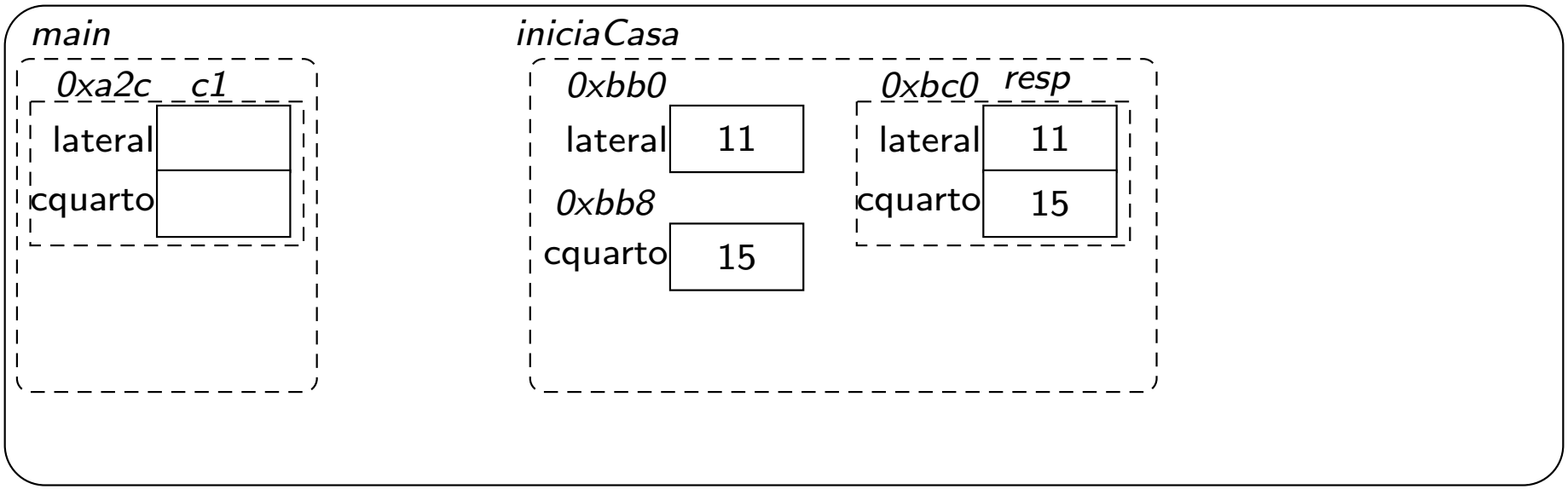


Estruturas e Memória

```
int main() {
  casa c1 = iniciaCasa(11,15);

  return 0;
}
```

```
casa iniciaCasa(float lateral, float cquarto) {
  casa resp;
  resp.lateral = lateral;
  resp.cquarto = cquarto;
  return resp;
}
```

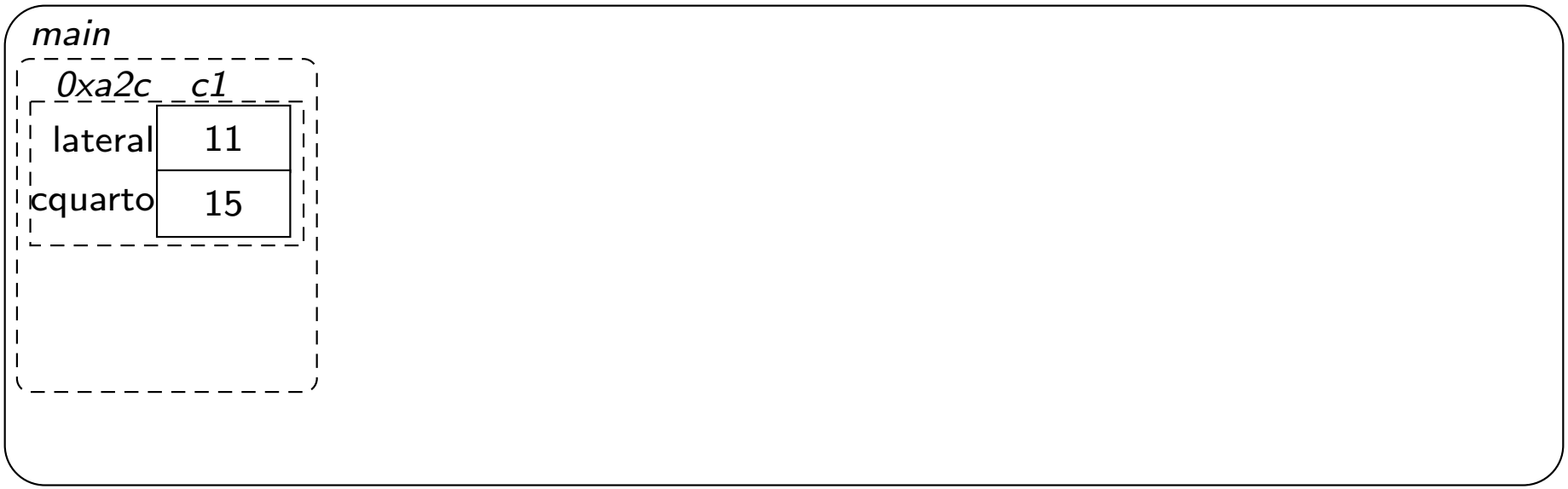


Estruturas e Memória

```
int main() {
  casa c1 = iniciaCasa(11,15);

  return 0;
}
```

```
casa iniciaCasa(float lateral, float cquarto) {
  casa resp;
  resp.lateral = lateral;
  resp.cquarto = cquarto;
  return resp;
}
```



Estruturas e Funções

- Observe o código ao lado

```
#include <stdio.h>

typedef struct auxCasa {
  float lateral;
  float cquarto;
} casa;

void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao: %.2f\n",ca.lateral);
}

int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;

  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

Estruturas e Funções

- Observe o código ao lado
- O que será impresso?

```
#include <stdio.h>

typedef struct auxCasa {
  float lateral;
  float cquarto;
} casa;

void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao: %.2f\n",ca.lateral);
}

int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;

  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

Estruturas e Funções

- Observe o código ao lado
- O que será impresso?

```c
#include <stdio.h>

typedef struct auxCasa {
  float lateral;
  float cquarto;
} casa;

void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao: %.2f\n",ca.lateral);
}

int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;

  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

Saída:

```
Lateral inicial: 11.00
Lateral na funcao: 12.00
Lateral final: 11.00
```

Estruturas e Memória

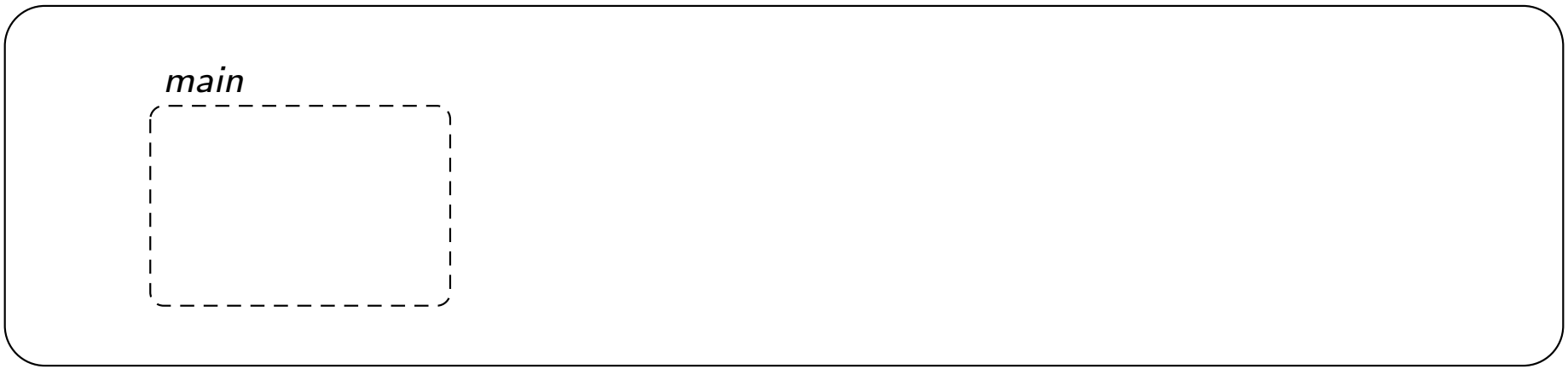
```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca.lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

Estruturas e Memória

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca.lateral);
}
```

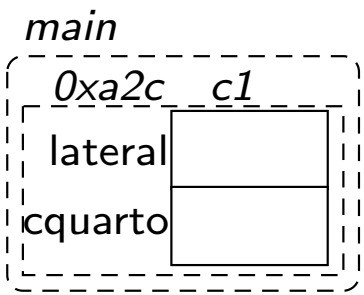
```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```



Estruturas e Memória

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca.lateral);
}
```

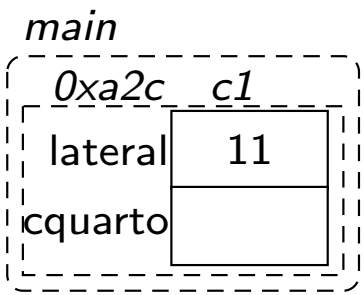
```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```



Estruturas e Memória

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca.lateral);
}
```

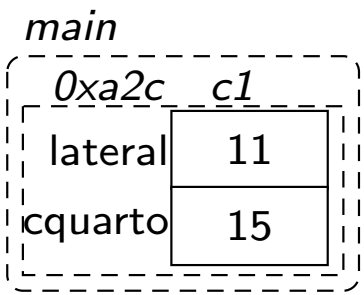
```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```



Estruturas e Memória

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca.lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

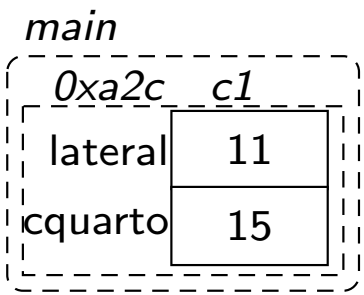


Estruturas e Memória

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca.lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
```

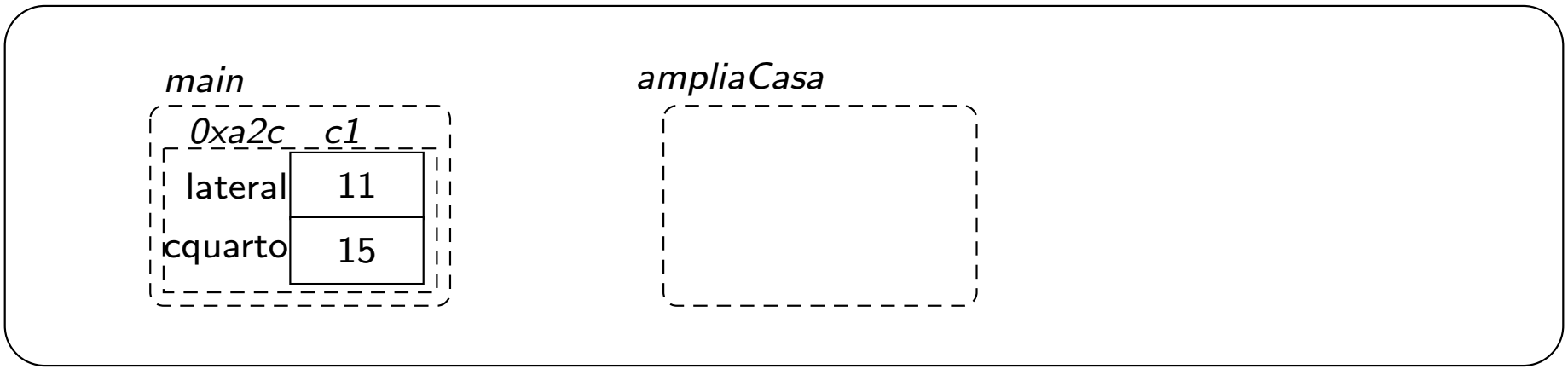


Estruturas e Memória

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca.lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
```

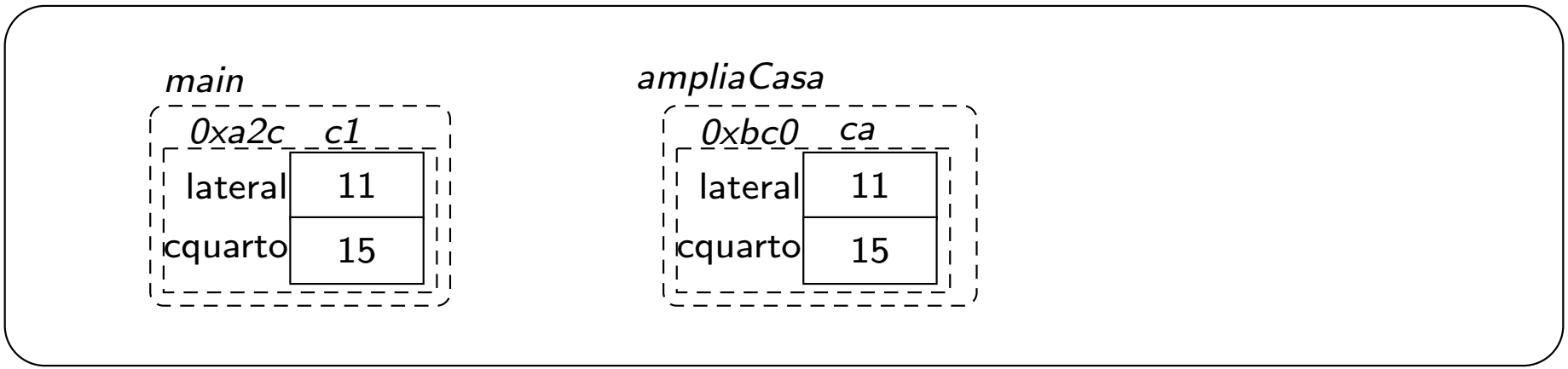


Estruturas e Memória

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca.lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
```

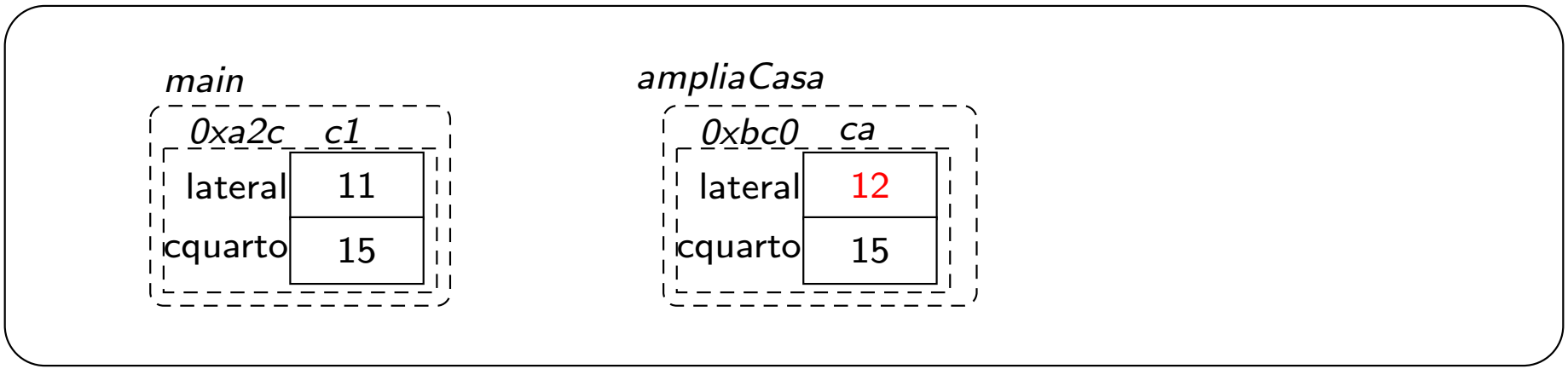


Estruturas e Memória

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca.lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
```

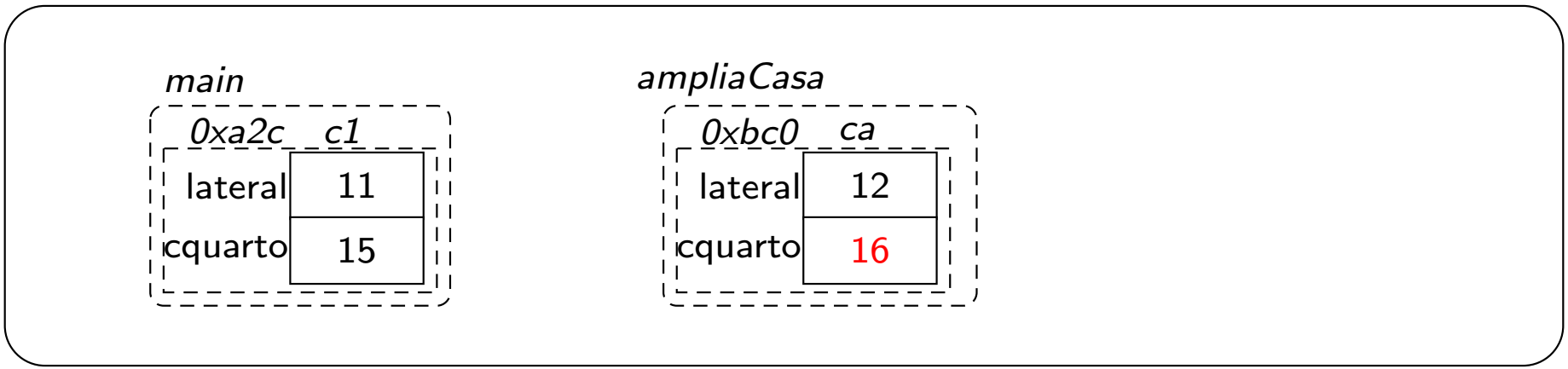


Estruturas e Memória

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca.lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
```

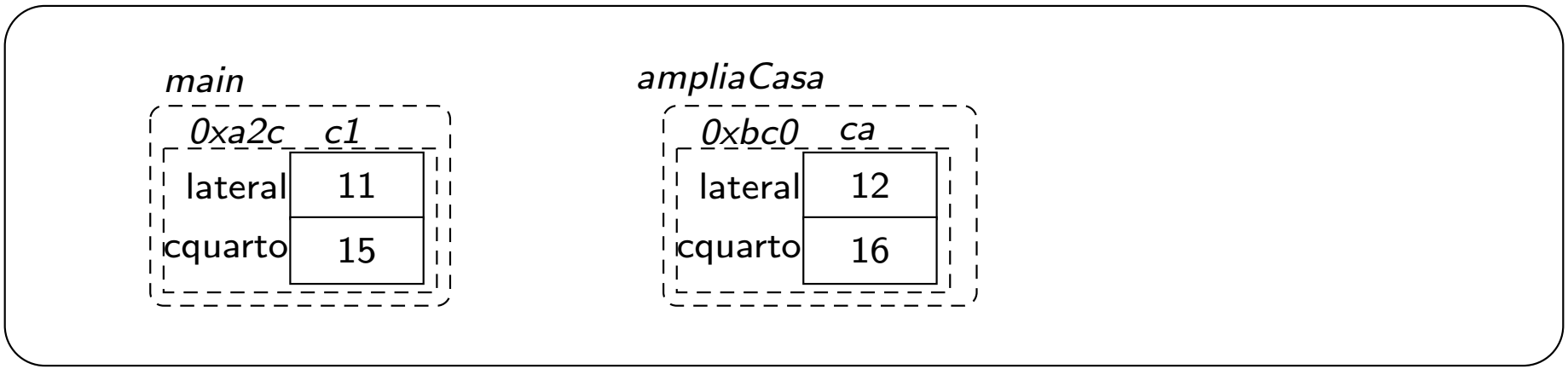


Estruturas e Memória

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca.lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
Lateral na funcao: 12.00
```

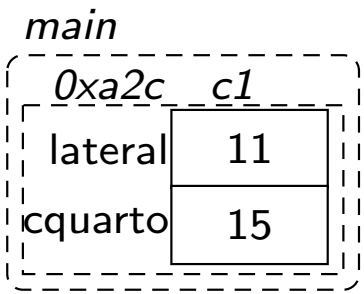


Estruturas e Memória

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca.lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
Lateral na funcao: 12.00
Lateral final: 11.00
```



Estruturas e Memória

- A variável `c1` foi passada como parâmetro

Estruturas e Memória

- A variável `c1` foi passada como parâmetro
  - Isto é, seu conteúdo de memória (o valor de seus campos) foi copiado para a função

Estruturas e Memória

- A variável `c1` foi passada como parâmetro
  - Isto é, seu conteúdo de memória (o valor de seus campos) foi copiado para a função
  - Chamamos isso de passagem de parâmetro por valor

Estruturas e Memória

- A variável c1 foi passada como parâmetro
  - Isto é, seu conteúdo de memória (o valor de seus campos) foi copiado para a função
  - Chamamos isso de passagem de parâmetro por valor
- Para podermos alterar os valores dos campos de c1 precisaríamos passar seu endereço

Estruturas e Memória

- A variável c1 foi passada como parâmetro
  - Isto é, seu conteúdo de memória (o valor de seus campos) foi copiado para a função
  - Chamamos isso de passagem de parâmetro por valor
- Para podermos alterar os valores dos campos de c1 precisaríamos passar seu endereço
  - Chamamos isso de passagem de parâmetro por referência

Estruturas e Memória

- Podemos passar o endereço de uma variável do tipo casa como referência a uma função

Estruturas e Memória

- Podemos passar o endereço de uma variável do tipo `casa` como referência a uma função
  - Este parâmetro seria do tipo `casa*`

Estruturas e Memória

- Podemos passar o endereço de uma variável do tipo `casa` como referência a uma função
  - Este parâmetro seria do tipo `casa*`
  - Exemplo: `void ampliaCasa2(casa* ca)`

```
void ampliaCasa2(casa* ca) {



}
```

Estruturas e Memória

- E como acessamos os campos de uma estrutura a partir de seu endereço?

```
void ampliaCasa2(casa* ca) {



}
```

Estruturas e Memória

- E como acessamos os campos de uma estrutura a partir de seu endereço?
  - Já aprendemos que podemos usar o * para acessar o conteúdo referenciado por um endereço de memória

```
void ampliaCasa2(casa* ca) {



}
```

Estruturas e Memória

- E como acessamos os campos de uma estrutura a partir de seu endereço?
  - Já aprendemos que podemos usar o * para acessar o conteúdo referenciado por um endereço de memória
  - Ex: `(*ca).lateral`

```
void ampliaCasa2(casa* ca) {
  (*ca).lateral++;


}
```

Estruturas e Memória

- E como acessamos os campos de uma estrutura a partir de seu endereço?
  - Já aprendemos que podemos usar o `*` para acessar o conteúdo referenciado por um endereço de memória
  - Ex: `(*ca).lateral`
  - Mas também podemos usar a "seta": `->`

```
void ampliaCasa2(casa* ca) {
  (*ca).lateral++;


}
```

Estruturas e Memória

- E como acessamos os campos de uma estrutura a partir de seu endereço?
  - Já aprendemos que podemos usar o `*` para acessar o conteúdo referenciado por um endereço de memória
  - Ex: `(*ca).lateral`
  - Mas também podemos usar a "seta": `->`
  - Ex: `ca->lateral` significando: vá à memória apontada por `ca` e acesse o campo `lateral`

```
void ampliaCasa2(casa* ca) {
  (*ca).lateral++;
  ca->cquarto++;
  printf("Lateral na funcao: %.2f\n",ca->lateral);
}
```

Estruturas e Memória

Passagem por valor

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao: %.2f\n",
                          ca.lateral);
}

int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",
                        c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",
                        c1.lateral);
  return 0;
}
```

Passagem por referência

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao: %.2f\n",
                          ca->lateral);
}

int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",
                        c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",
                        c1.lateral);
  return 0;
}
```

Estruturas e Memória

Passagem por valor

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao: %.2f\n",
                          ca.lateral);
}

int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",
                        c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",
                        c1.lateral);
  return 0;
}
```

Passagem por referência

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao: %.2f\n",
                          ca->lateral);
}

int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",
                        c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",
                        c1.lateral);
  return 0;
}
```

Estruturas e Memória

Passagem por valor

```
void ampliaCasa(casa ca) {
  ca.lateral++;
  ca.cquarto++;
  printf("Lateral na funcao: %.2f\n",
                          ca.lateral);
}

int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",
                        c1.lateral);
  ampliaCasa(c1);
  printf("Lateral final: %.2f\n",
                        c1.lateral);
  return 0;
}
```

Passagem por referência

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao: %.2f\n",
                          ca->lateral);
}

int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",
                        c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",
                        c1.lateral);
  return 0;
}
```

Estruturas e Memória

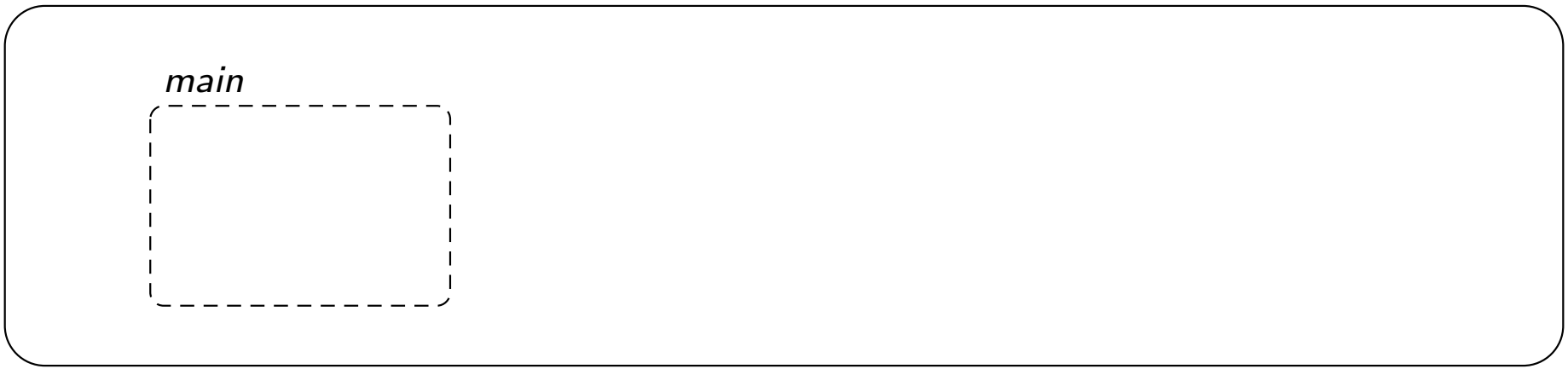
```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca->lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

Estruturas e Memória

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca->lateral);
}
```

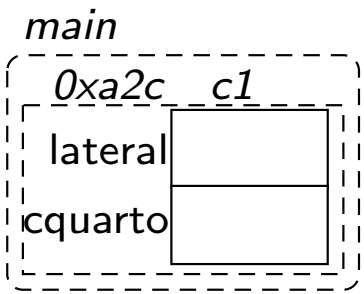
```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```



Estruturas e Memória

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca->lateral);
}
```

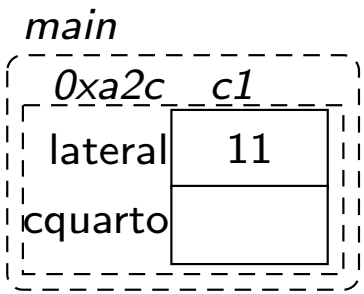
```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```



Estruturas e Memória

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca->lateral);
}
```

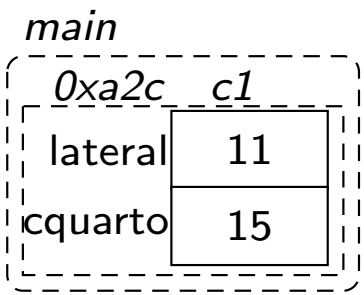
```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```



Estruturas e Memória

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca->lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

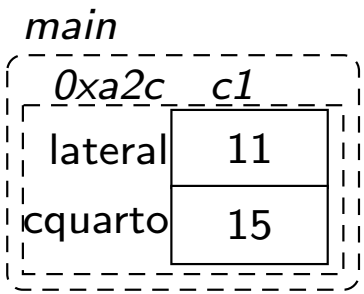


Estruturas e Memória

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca->lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
```

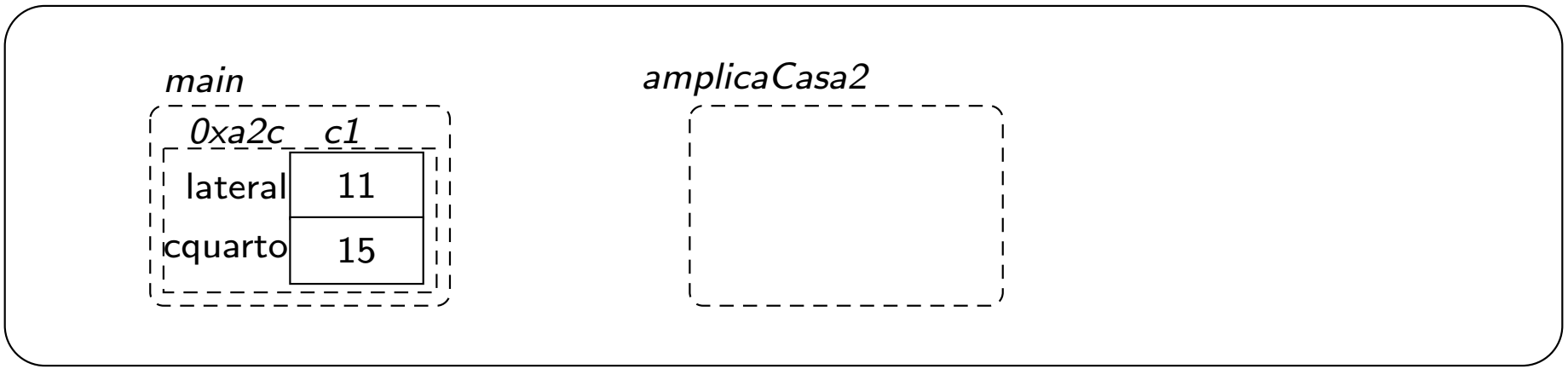


Estruturas e Memória

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca->lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
```

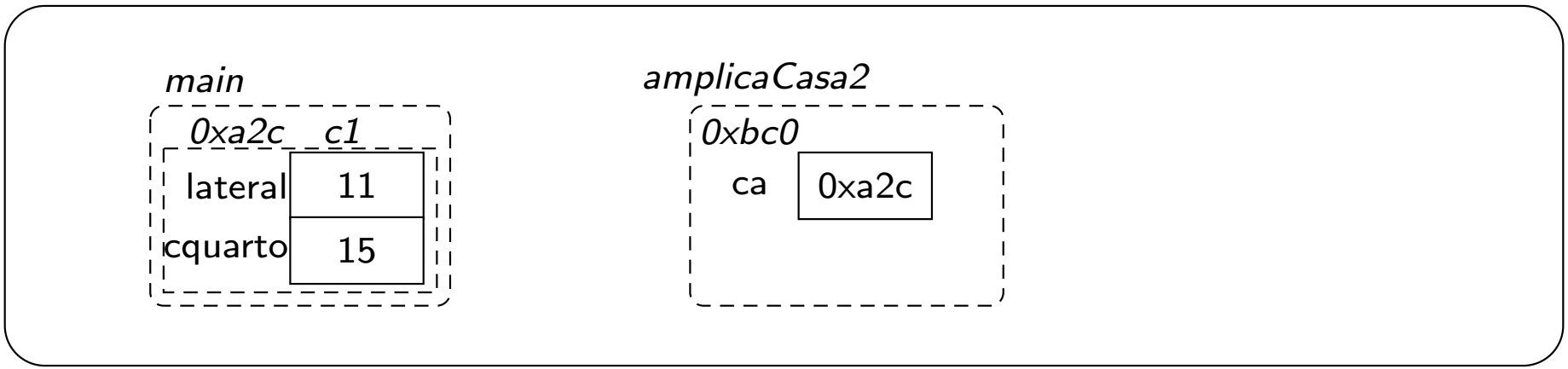


Estruturas e Memória

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca->lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
```

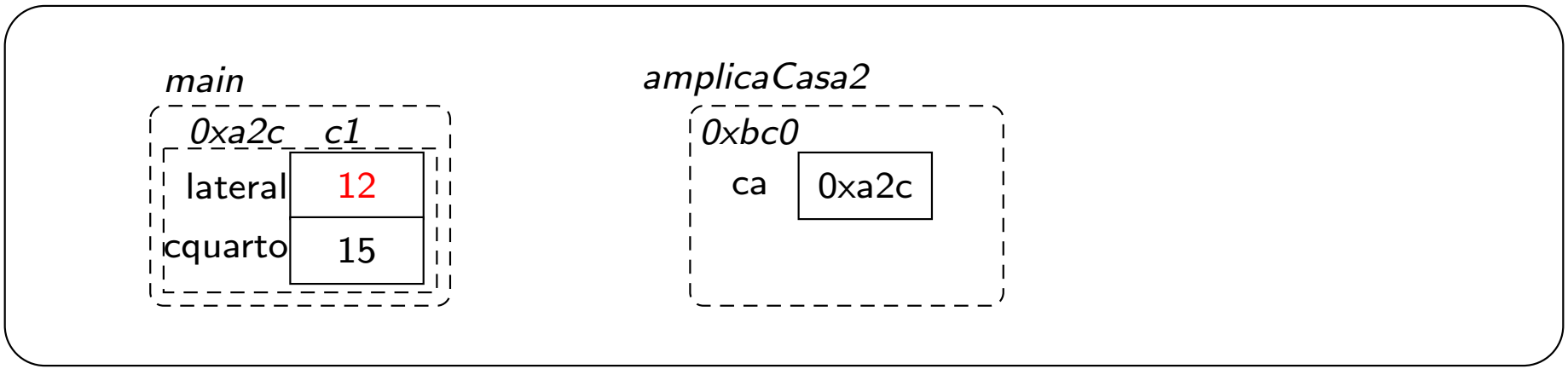


Estruturas e Memória

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca->lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
```

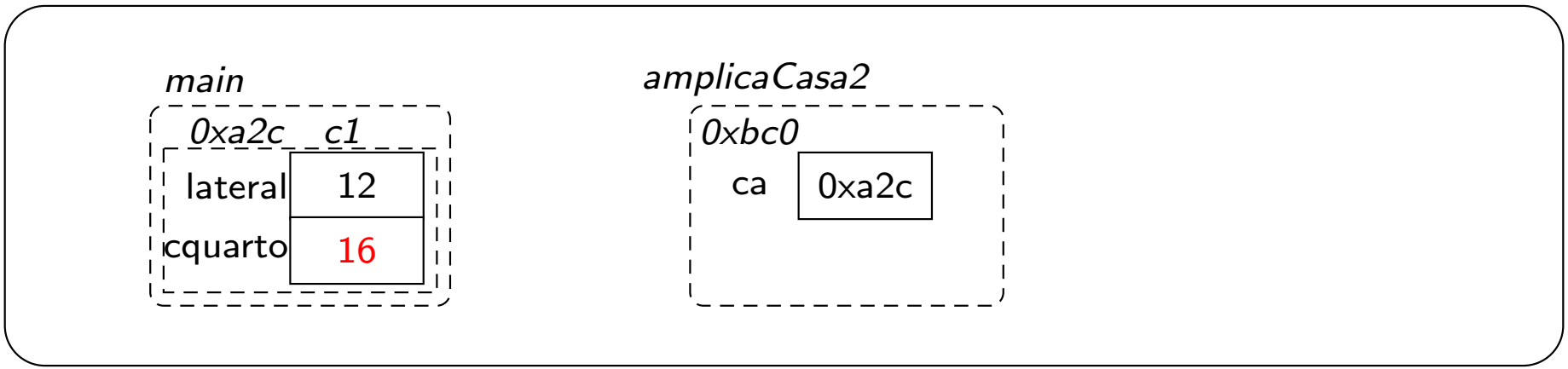


Estruturas e Memória

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca->lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
```

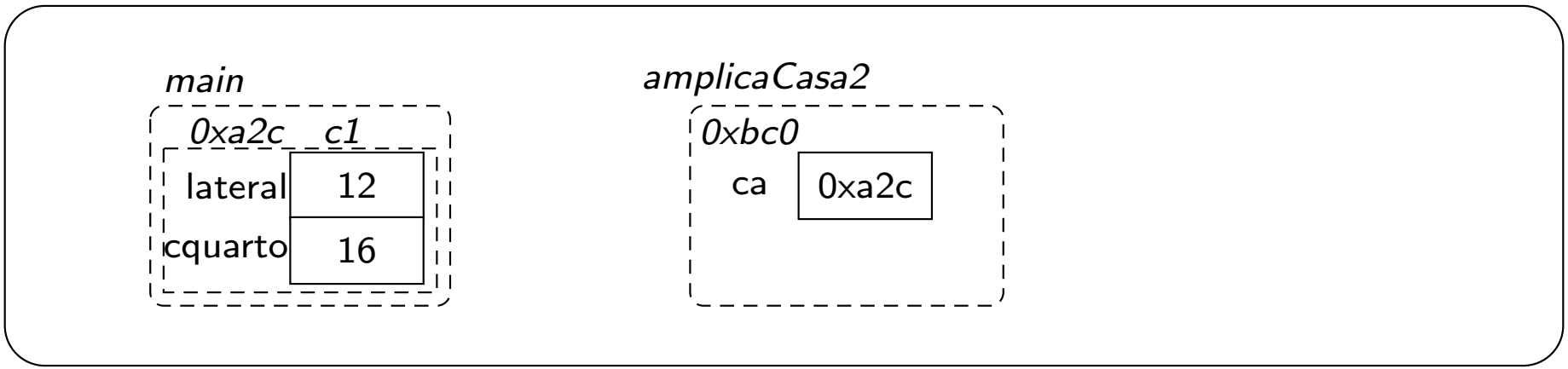


Estruturas e Memória

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca->lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
Lateral na funcao: 12.00
```

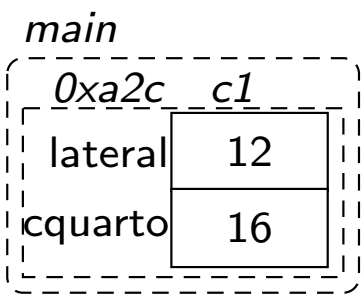


Estruturas e Memória

```
void ampliaCasa2(casa* ca) {
  ca->lateral++;
  ca->cquarto++;
  printf("Lateral na funcao:
         %.2f\n",ca->lateral);
}
```

```
int main() {
  casa c1;
  c1.lateral = 11;
  c1.cquarto = 15;
  printf("Lateral inicial: %.2f\n",c1.lateral);
  ampliaCasa2(&c1);
  printf("Lateral final: %.2f\n",c1.lateral);
  return 0;
}
```

```
Lateral inicial: 11.00
Lateral na funcao: 12.00
Lateral final: 12.00
```



Aula 26 – Estruturas (parte 2)

Norton T. Roman & Luciano A. Digiampietri