TDDE18 & 726G77

Pointers & Dynamic memory

Christoffer Holm

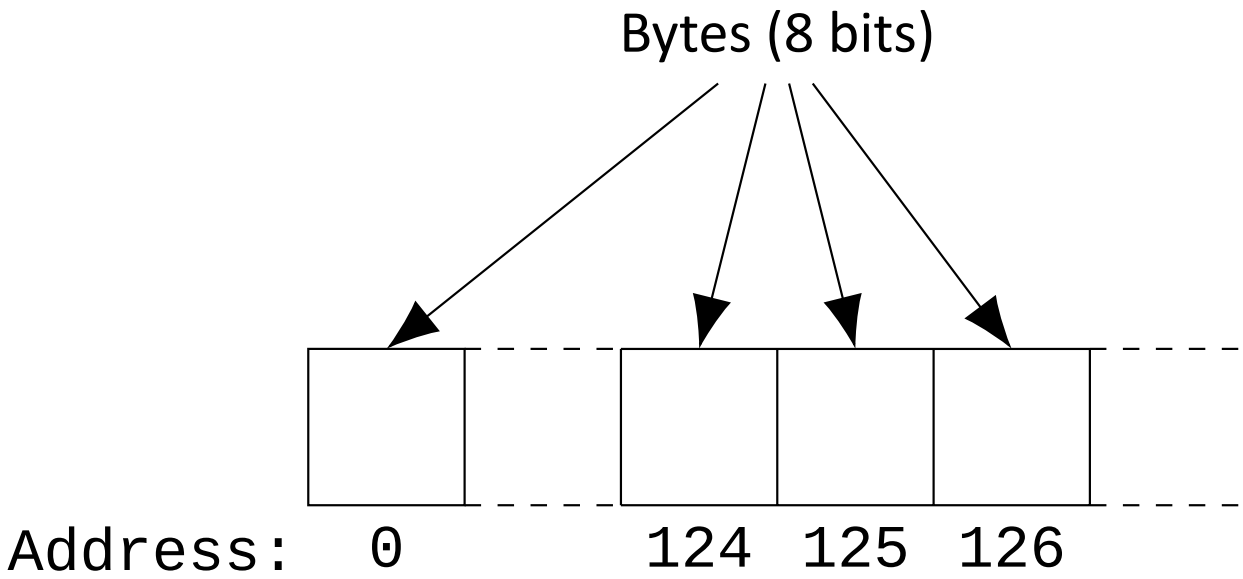
Department of Computer and information science

LINKÖPING UNIVERSITY

1 Memory
2 Pointers
3 Dynamic Memory
4 Structs & Pointers

Memory

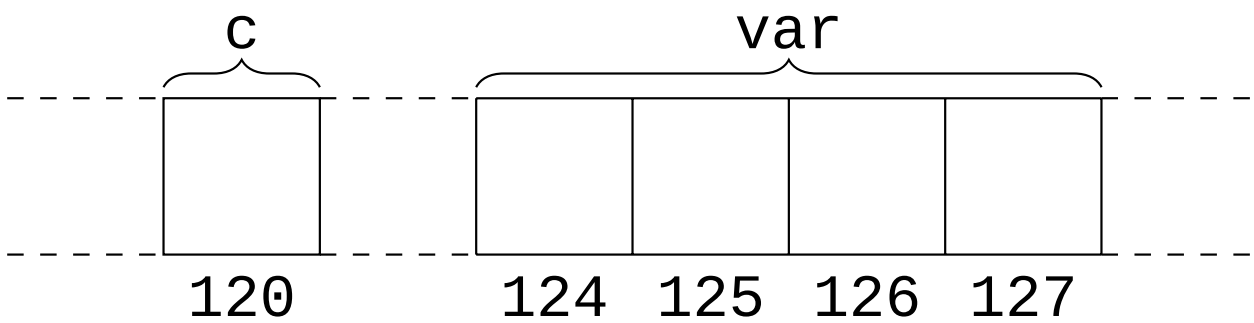
What is memory?



Memory

So how are variables stored then?

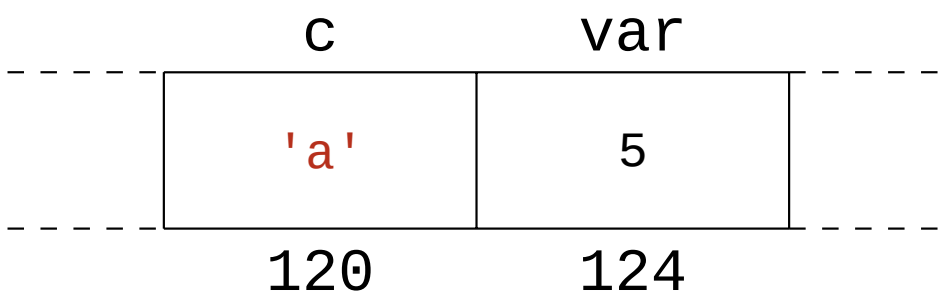
```
char c { 'a' }; // takes 1 byte
int var { 5 };  // takes 4 bytes
```



Memory

So how are variables stored then?

```
char c { 'a' };
int var { 5 };
```



Memory

The address and size of a variable

`&var` = The address of `var`

`sizeof(var)` = The number of bytes in `var`

LINKÖPING UNIVERSITY

Pointers

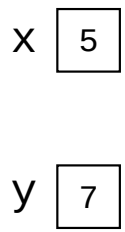
What is a pointer?

```
int x{5};
int y{7};
```

Pointers

What is a pointer?

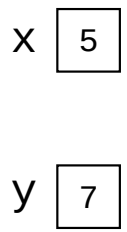
```
int x{5};
int y{7};
```



Pointers

What is a pointer?

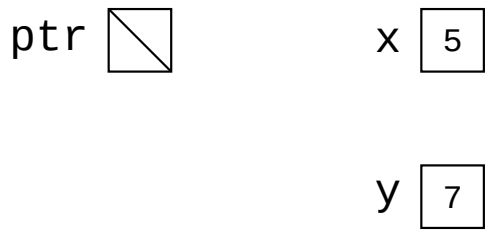
```
int x{5};
int y{7};
int* ptr{};
```



Pointers

What is a pointer?

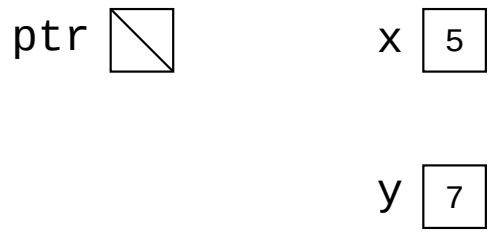
```
int x{5};
int y{7};
int* ptr{};
```



Pointers

What is a pointer?

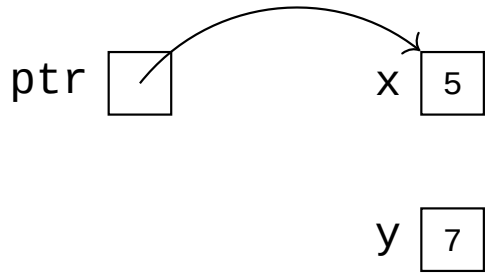
```
int x{5};
int y{7};
int* ptr{&x};
```



Pointers

What is a pointer?

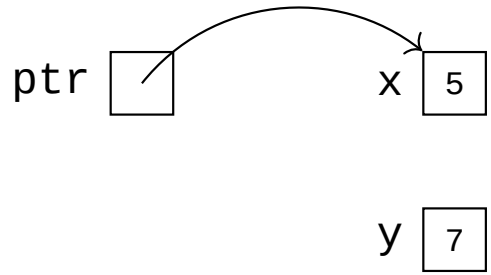
```
int x{5};
int y{7};
int* ptr{&x};
```



Pointers

What is a pointer?

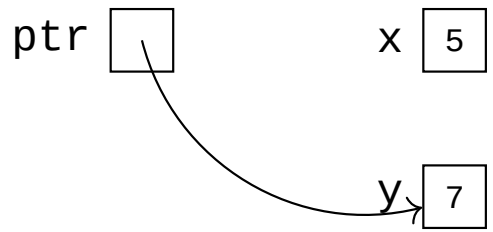
```
int x{5};
int y{7};
int* ptr{&x};
ptr = &y;
```



Pointers

What is a pointer?

```
int x{5};
int y{7};
int* ptr{&x};
ptr = &y;
```

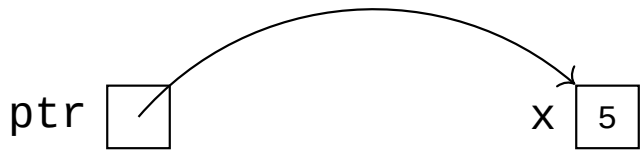


Pointers

How to use a pointer

```
int x{5};
int *ptr{&x};

cout << *ptr << endl;
```

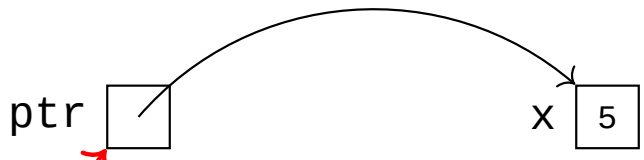


Pointers

How to use a pointer

```
int x{5};
int *ptr{&x};

cout << *ptr << endl;
```

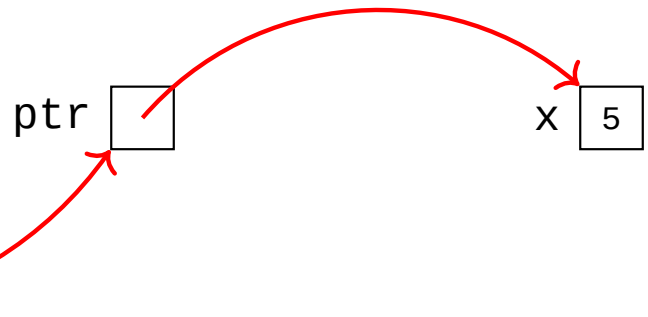


Pointers

How to use a pointer

```
int x{5};
int *ptr{&x};

cout << *ptr << endl;
```



Pointers

& and * are inverses to each other

```
int x { 17 };
int* ptr { &x };

// what does this print?
cout << *&x << endl;

// How about this?
cout << &*ptr << endl;
```

Pointers

Address of pointers

```
int x { 17 };
int* ptr { &x };

cout << *ptr; // prints '17'
cout << ptr;  // prints '0x120'
cout << &ptr; // prints '0x124'
```

Pointers

Pointing to nothing

```
int* ptr{};
ptr == nullptr;
```

1 Memory
2 Pointers
3 Dynamic Memory
4 Structs & Pointers

LINKÖPING UNIVERSITY

Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```

Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```

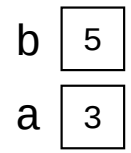
Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```



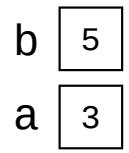
Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```



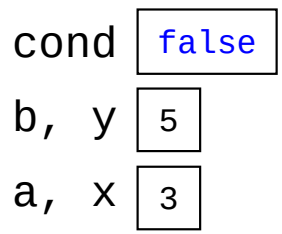
Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```



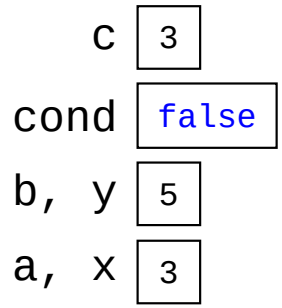
Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```



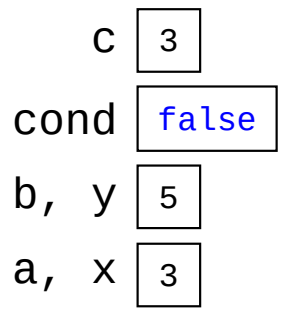
Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```



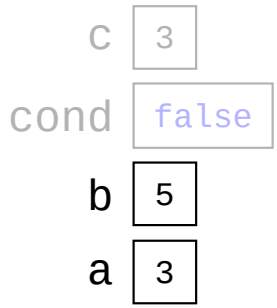
Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```



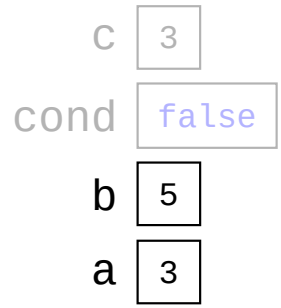
Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```



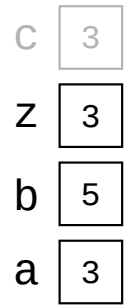
Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```



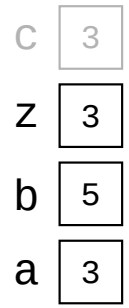
Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```



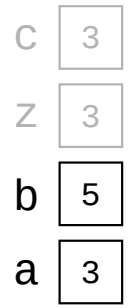
Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```



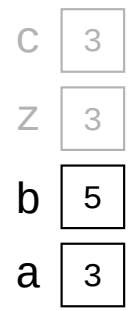
Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```



Dynamic Memory

How are variables managed in memory?

```
void swap_if(int& x, int& y, bool cond) {
  int c { x };
  if (cond) {
    x = y;
    y = c;
  }
}

void print(int z) {
  cout << z << endl;
}

int main() {
  int a {3};
  int b {5};
  swap_if(a, b, a > b);
  print(a);
}
```

Dynamic Memory

What goes wrong in this example?

```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```

Dynamic Memory

What goes wrong in this example?

```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```

Dynamic Memory

What goes wrong in this example?

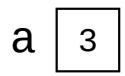
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

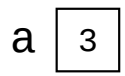
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

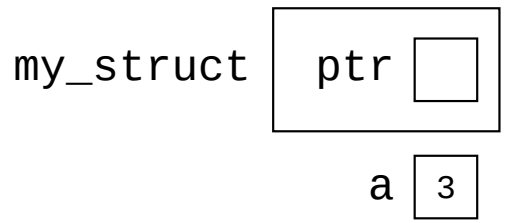
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

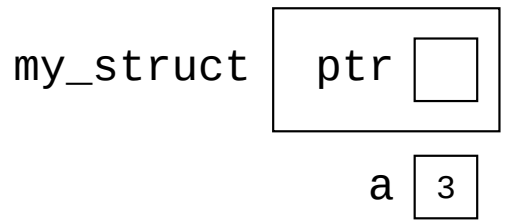
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

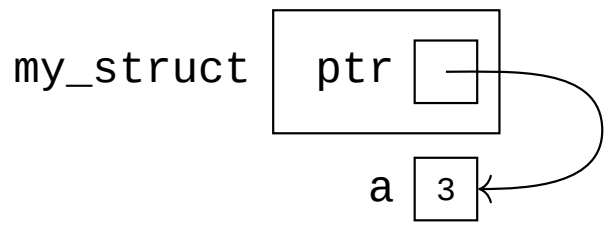
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

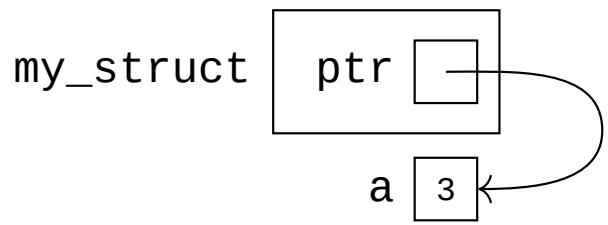
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

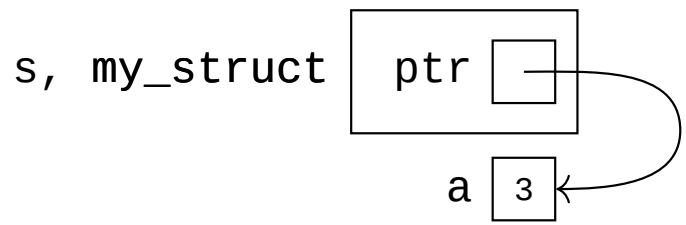
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

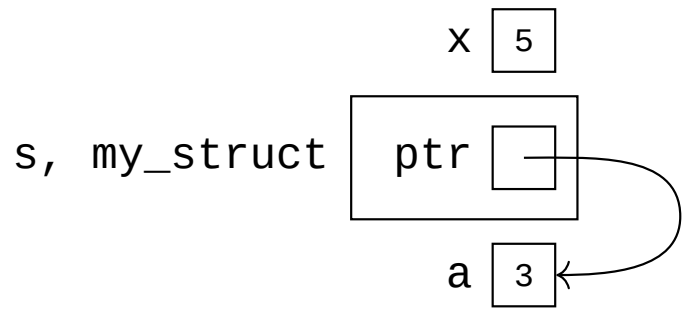
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

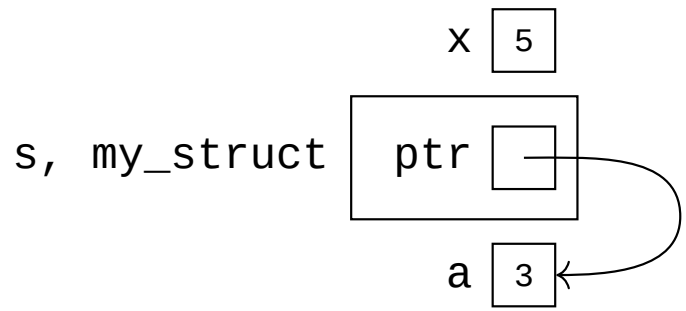
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

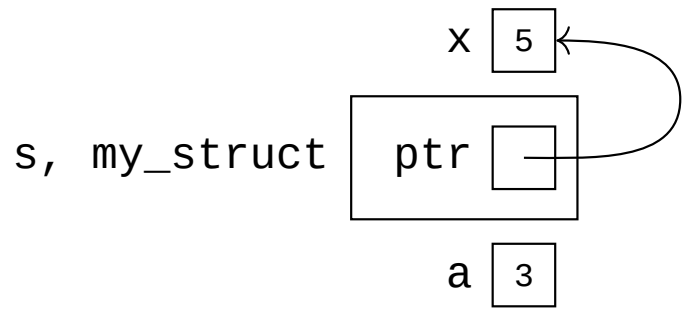
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

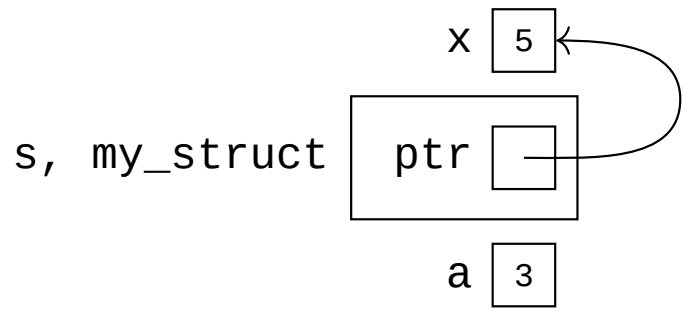
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

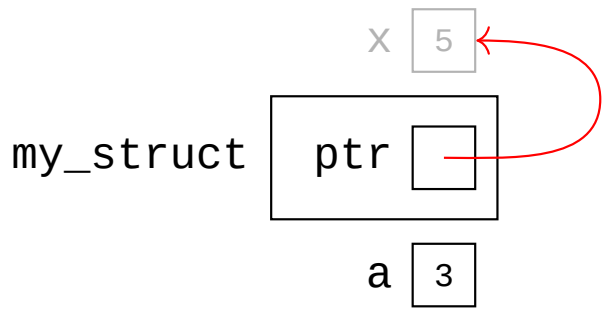
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

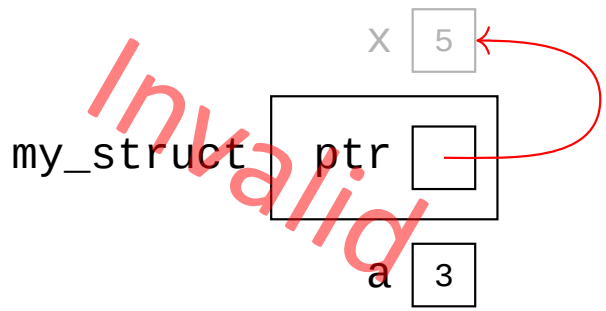
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

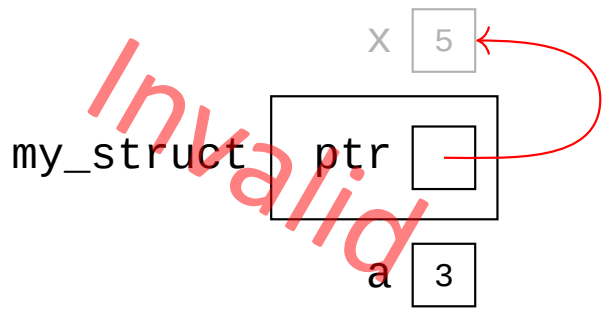
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

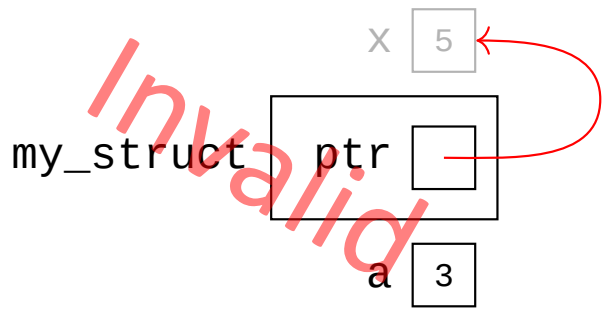
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What goes wrong in this example?

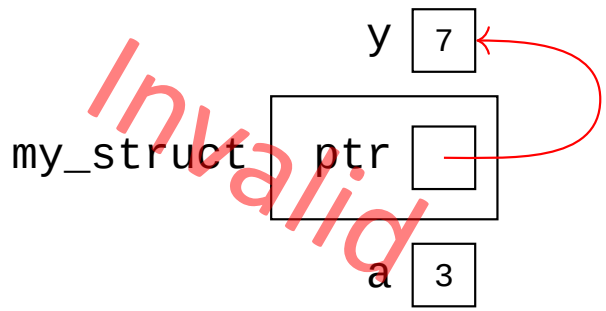
```
struct My_Struct { int* ptr; };

void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = &x;
}

void do_stuff() {
  int y { 7 };
  // ...
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  do_stuff();
}
```



Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```

Heap

- - - - - - - - - - - - - - - - - - -

Stack

Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```

Heap

Stack

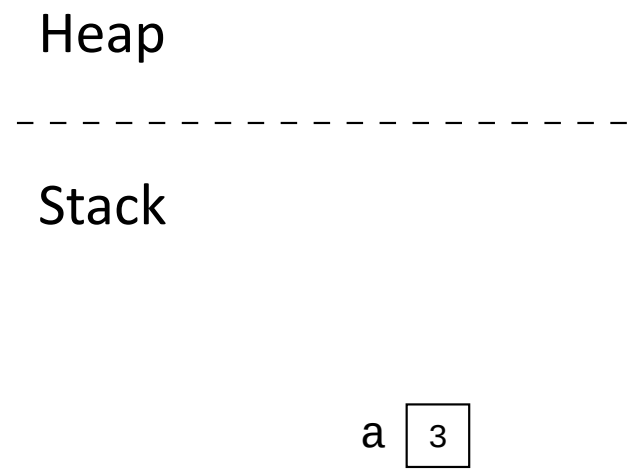
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



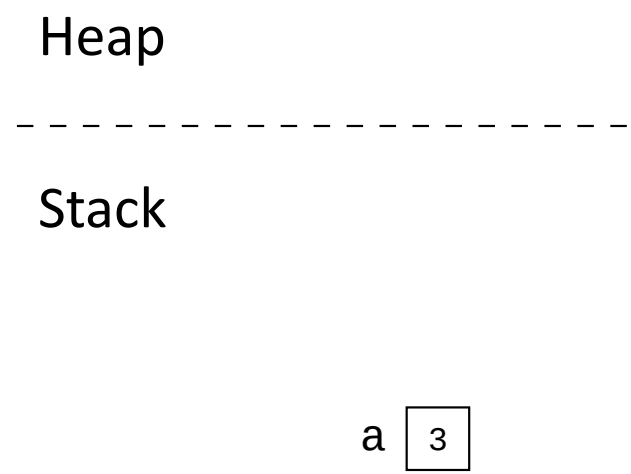
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



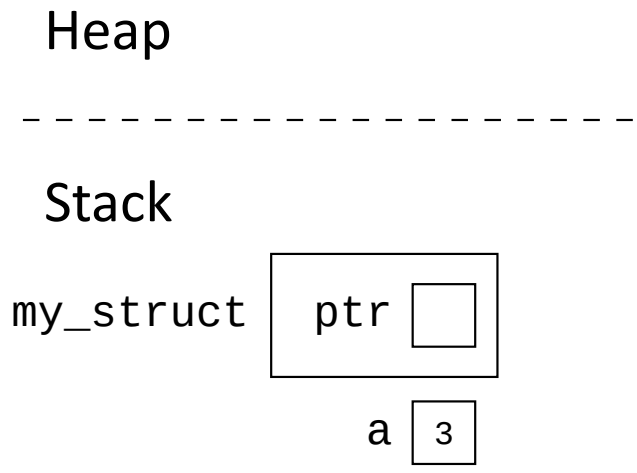
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



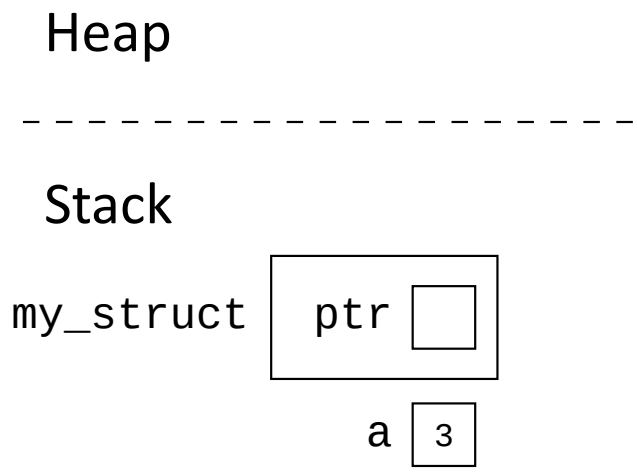
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



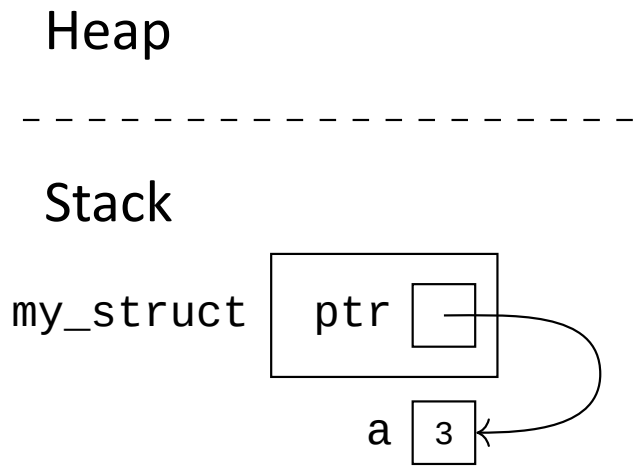
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



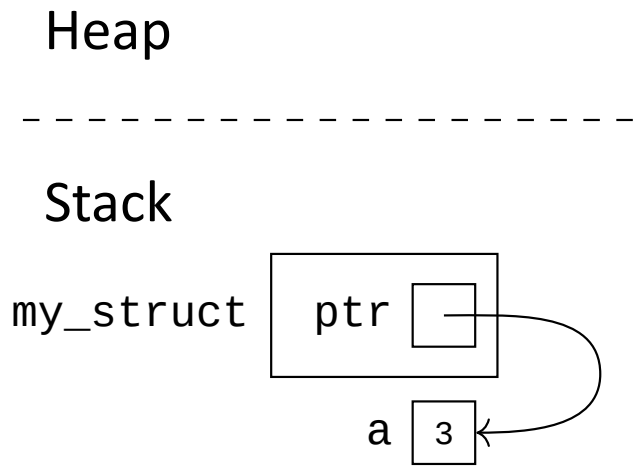
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



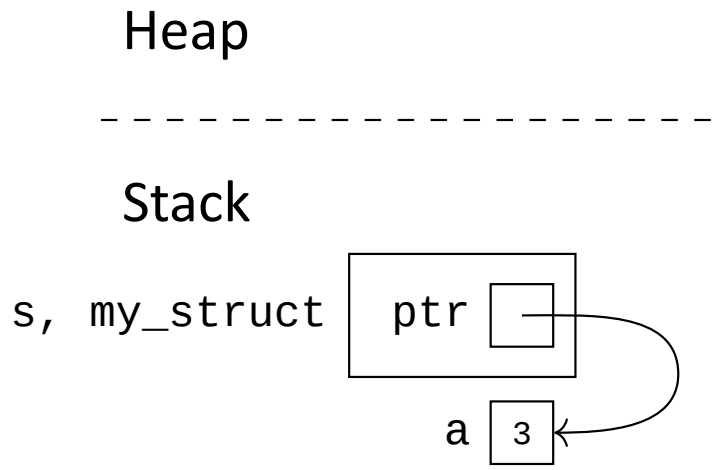
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



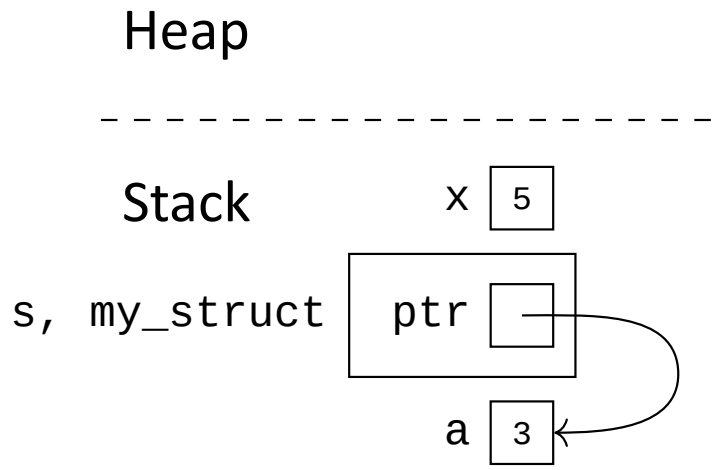
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



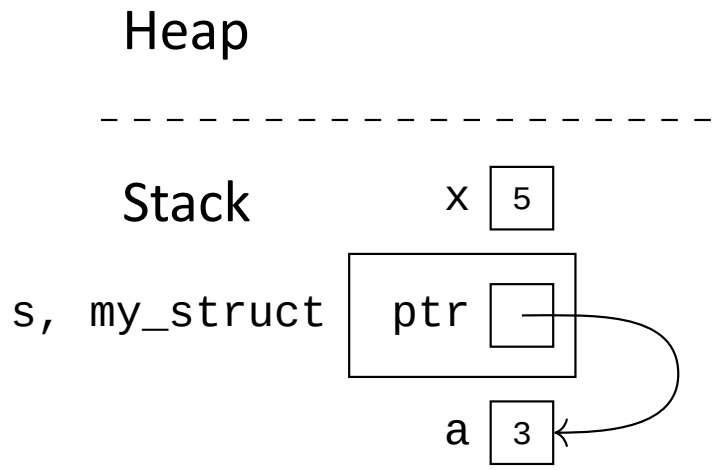
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



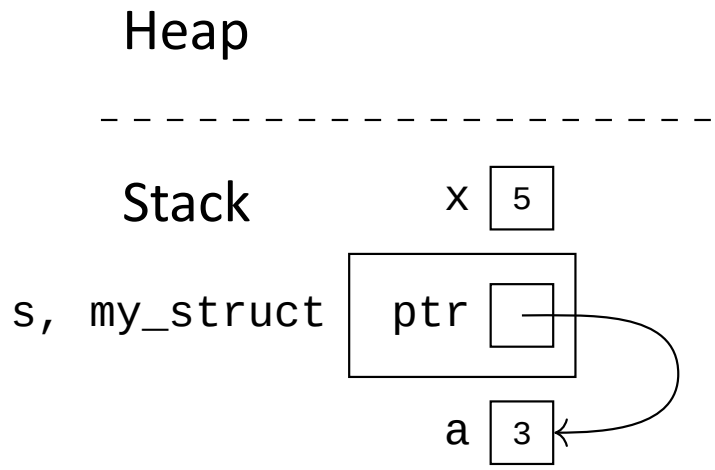
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



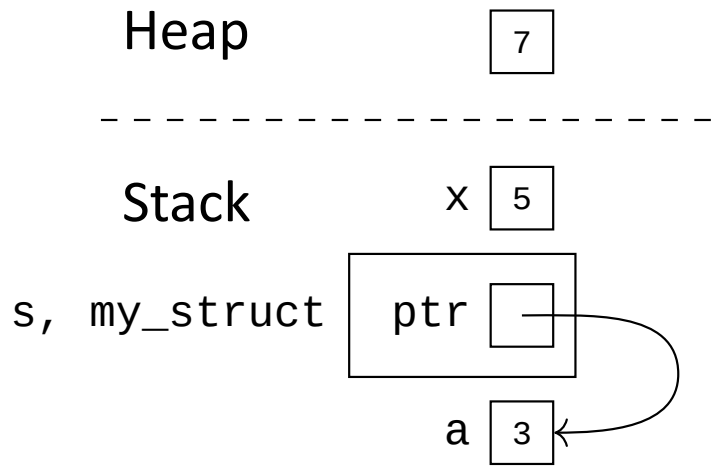
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



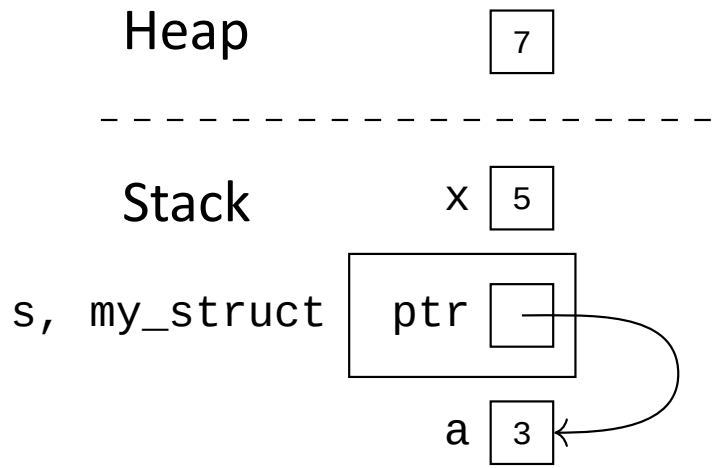
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



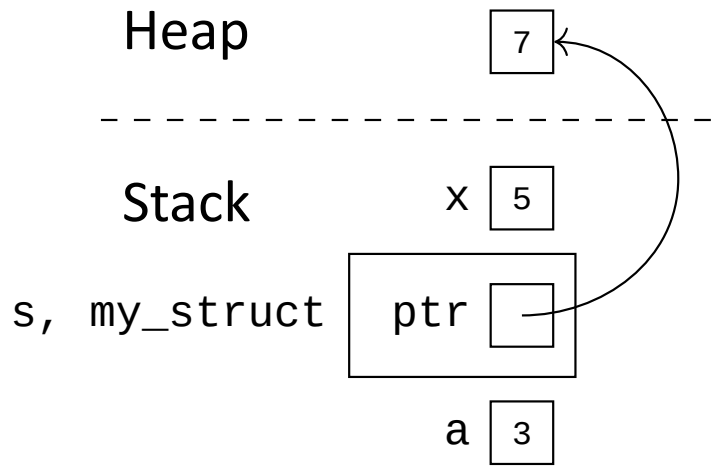
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



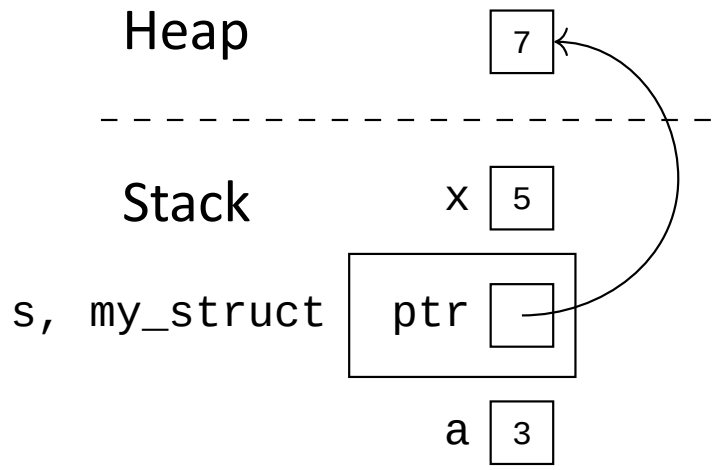
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



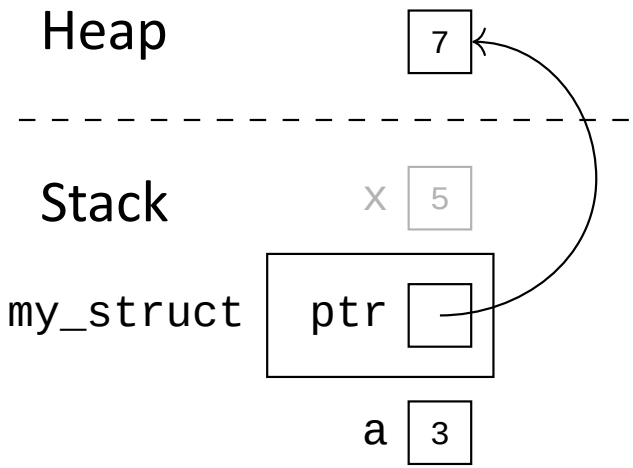
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



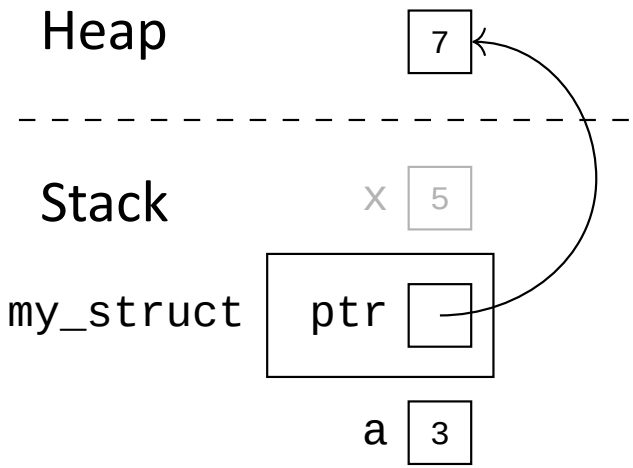
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



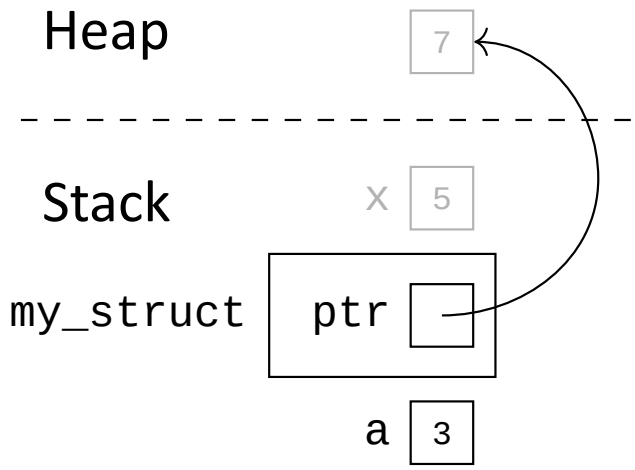
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```



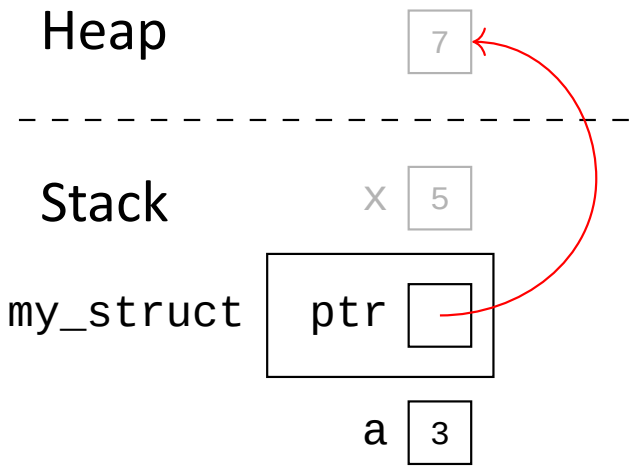
Dynamic Memory

What if we can store x somewhere else?

```
void change_pointer(My_Struct& s) {
  int x { 5 };
  s.ptr = new int{7};
}

int main() {
  int a { 3 };
  My_Struct my_struct { };

  my_struct.ptr = &a;
  change_pointer(my_struct);
  delete my_struct.ptr;
}
```

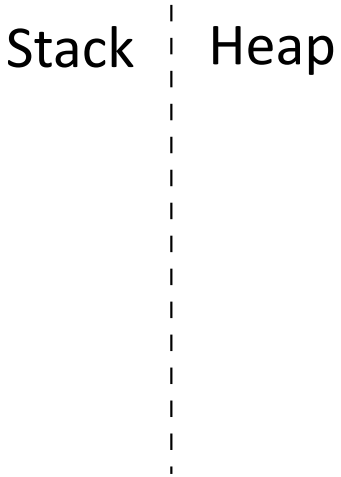


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

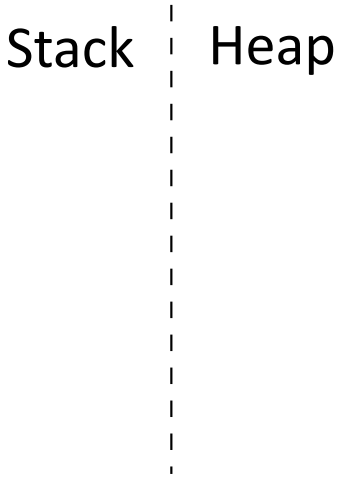


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

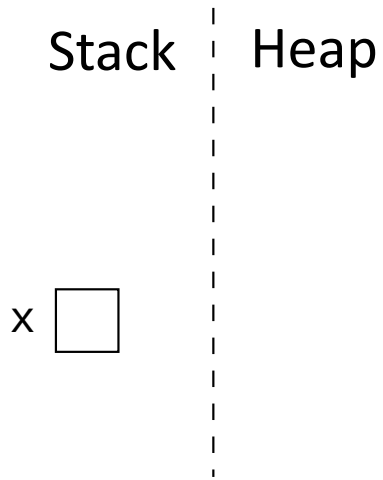


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

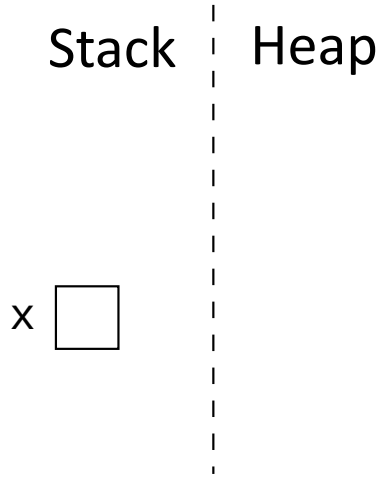


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

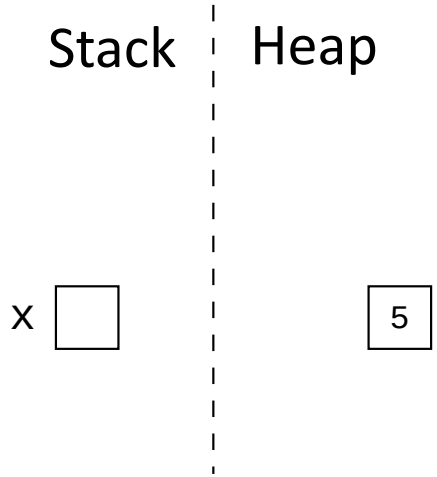


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

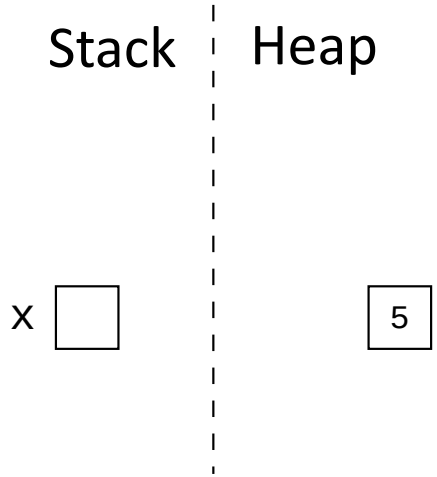


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

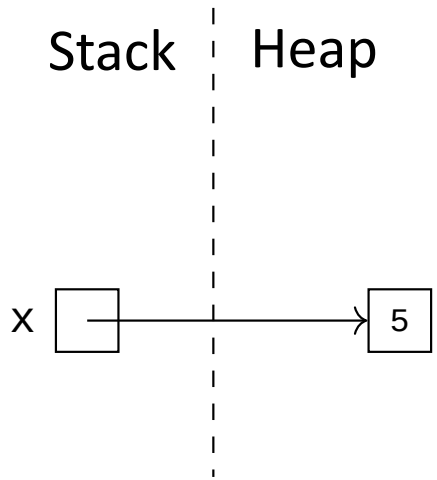


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

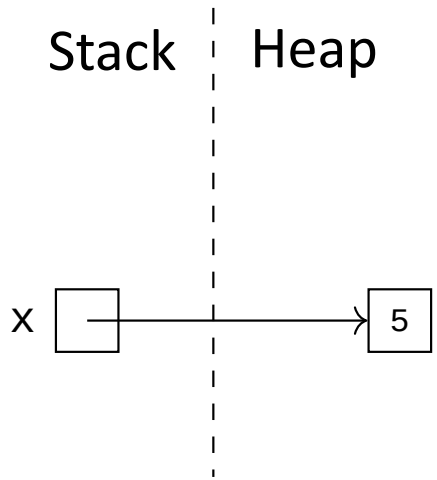


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

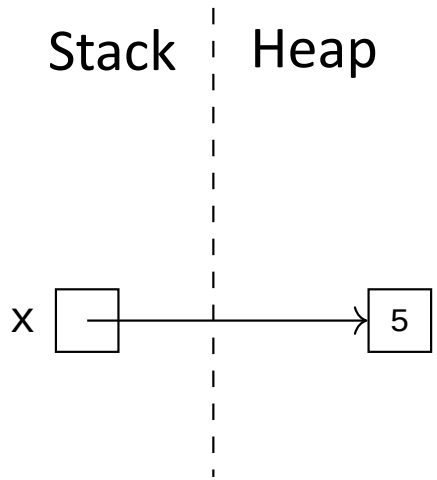


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

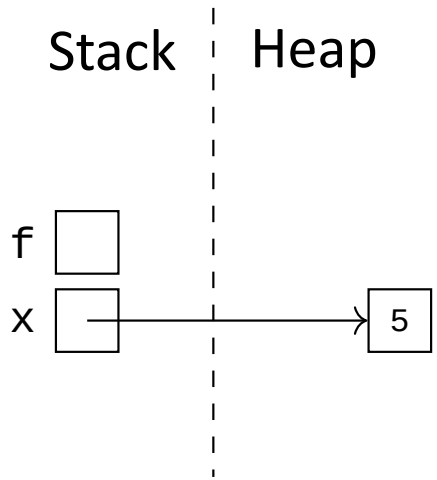


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

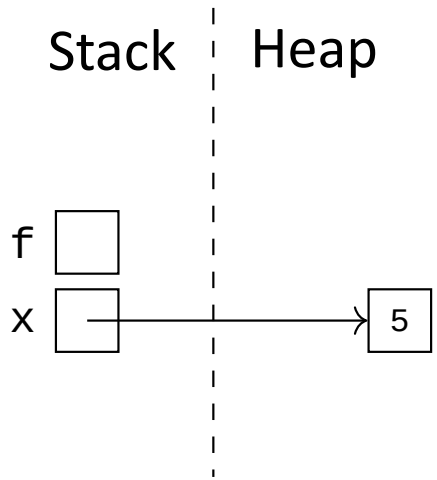


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

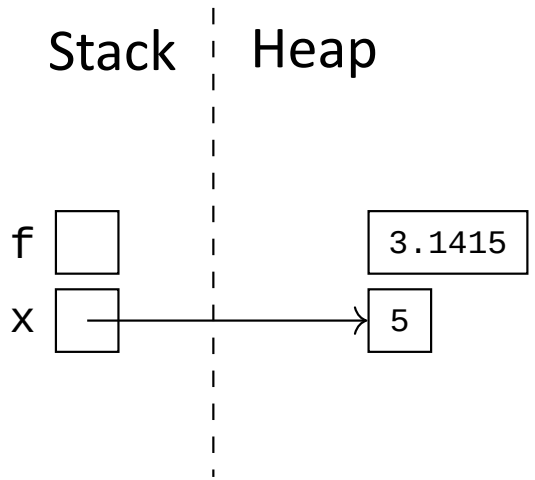


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

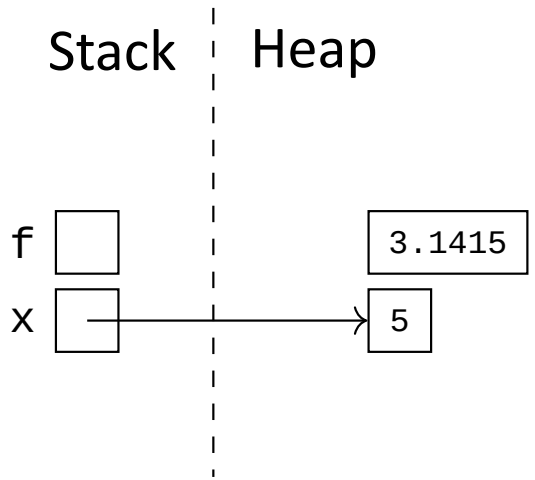


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

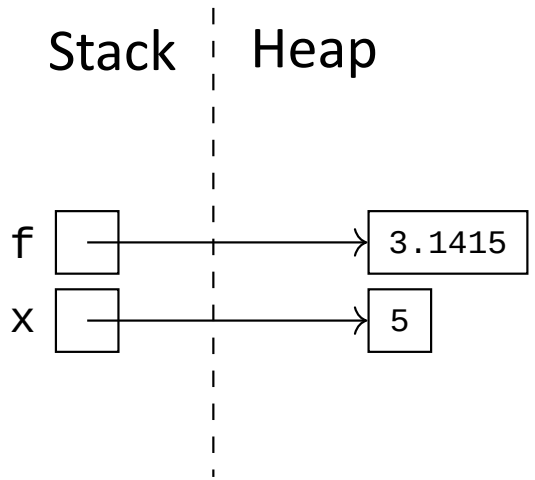


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

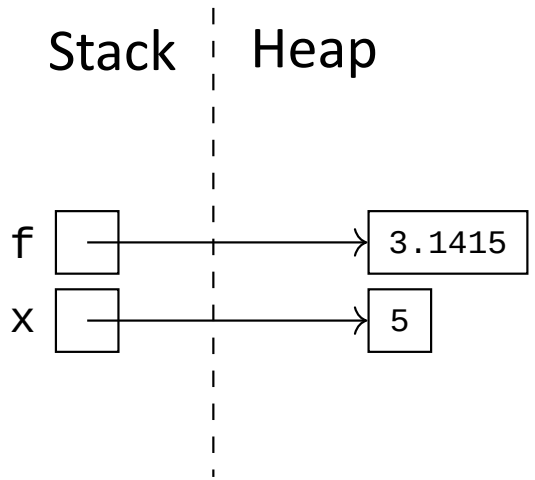


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

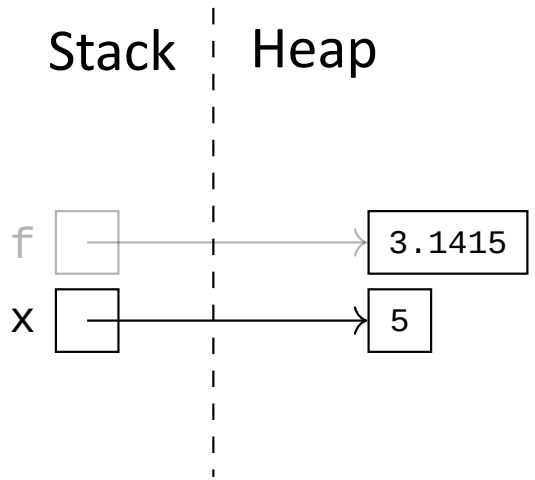


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

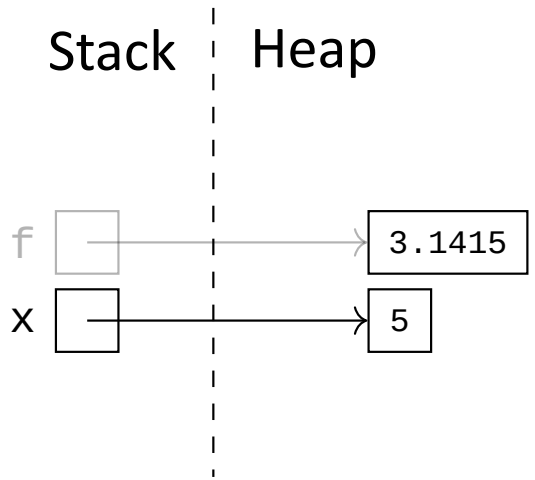


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

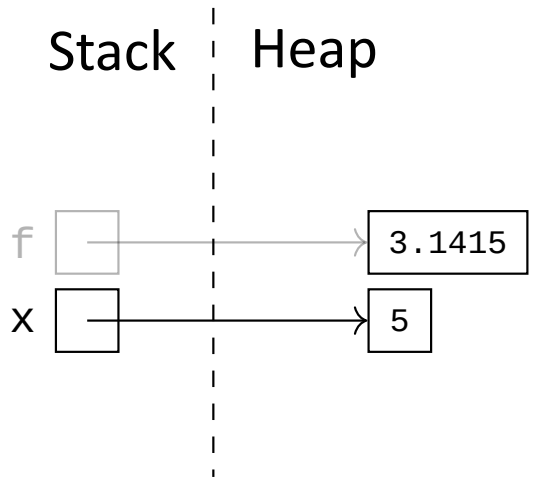


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

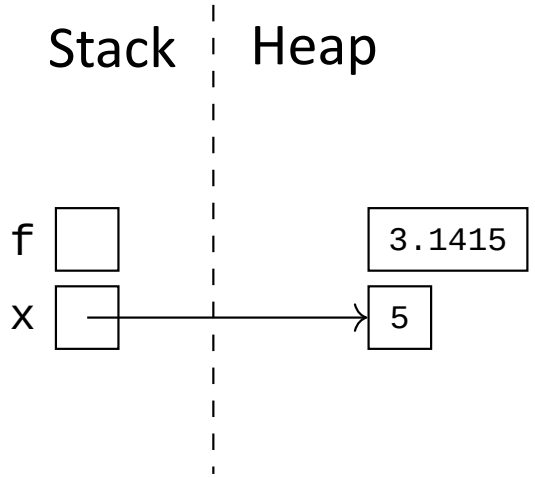


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

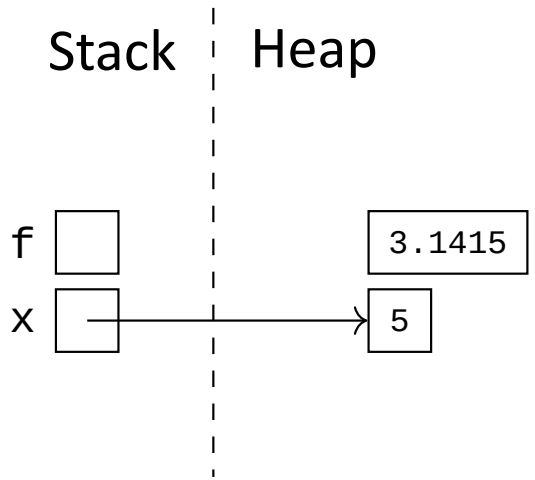


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

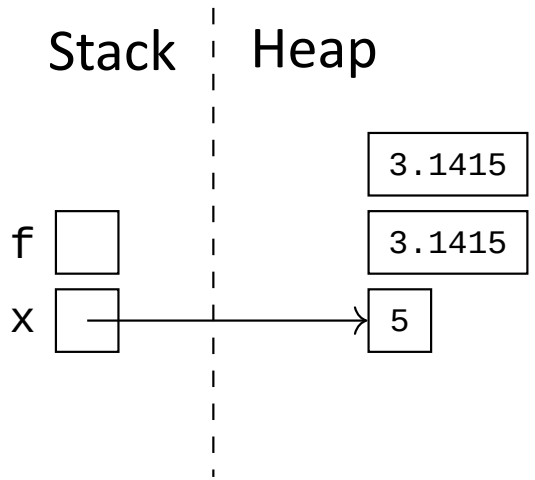


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

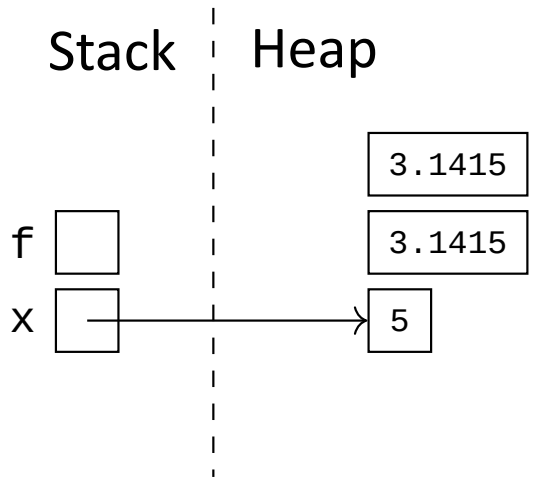


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

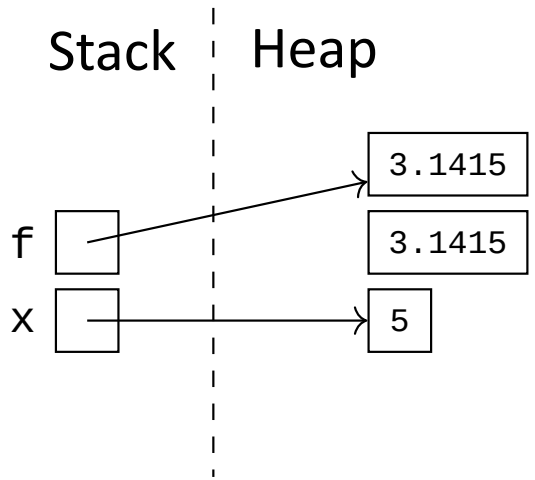


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

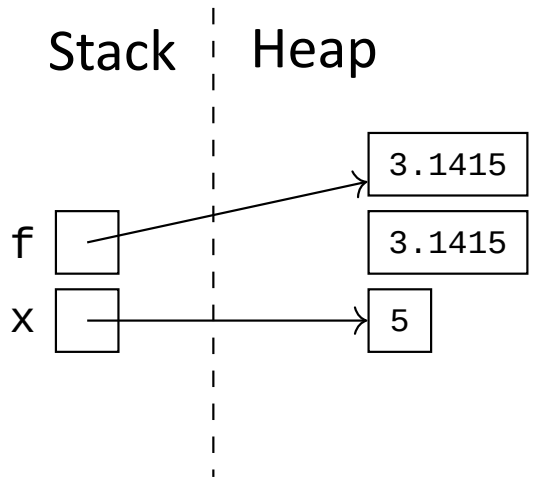


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

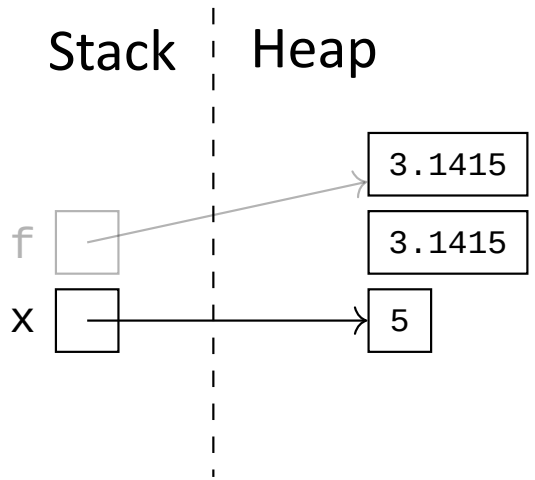


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

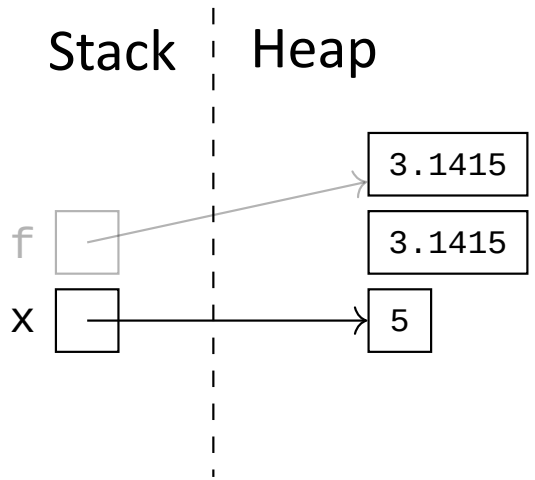


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

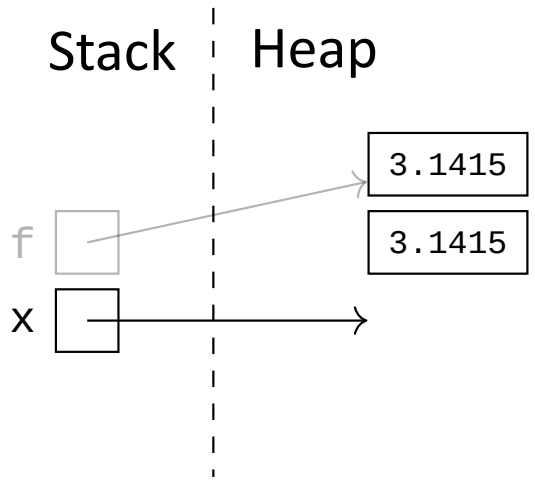


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

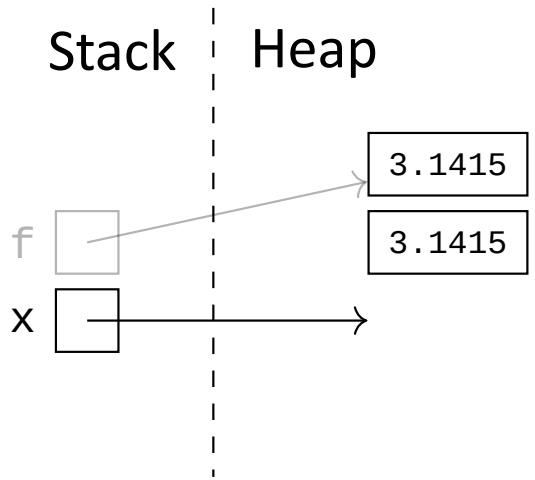


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

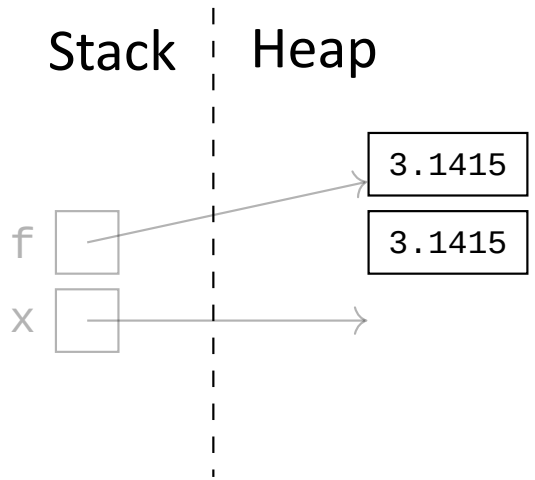


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

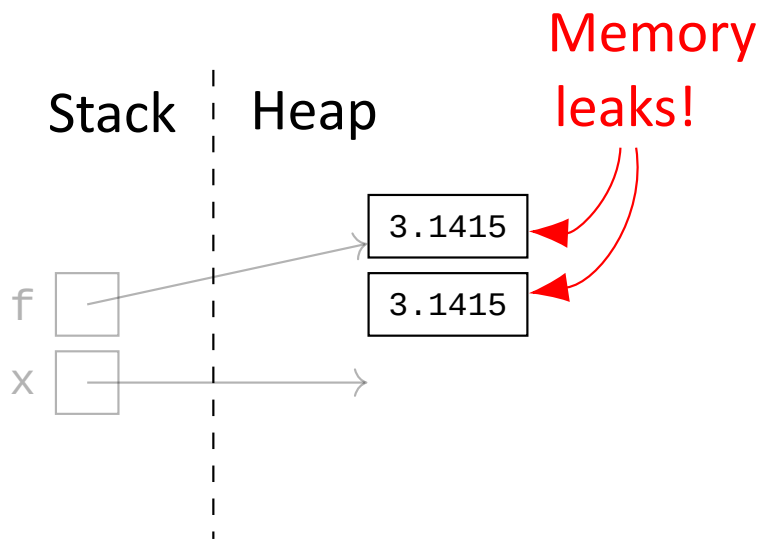


Dynamic Memory

Memory Leaks

```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```



Dynamic Memory

How to fix memory leaks?

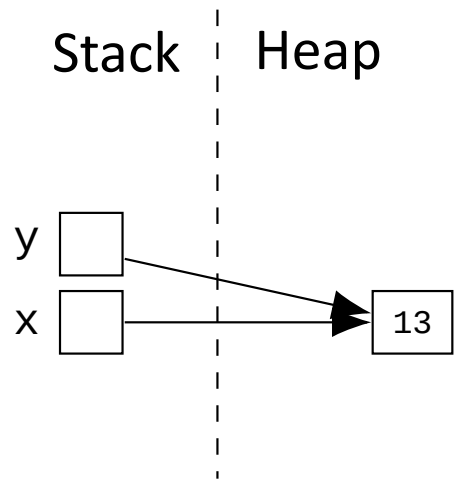
```
void do_stuff()
{
  float* f { new float{3.1415} };
  // ...
  delete f; // one delete for each new
}

int main()
{
  int* x { new int{5} };
  do_stuff();
  do_stuff();
  delete x;
}
```

Dynamic Memory

Double delete

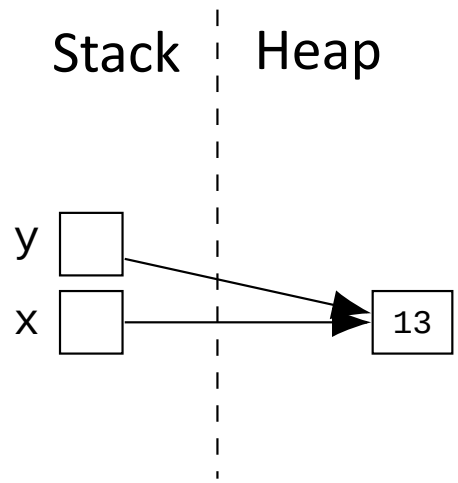
```
int* x { new int{13} };
int* y { x };
delete x;
delete y;
```



Dynamic Memory

Double delete

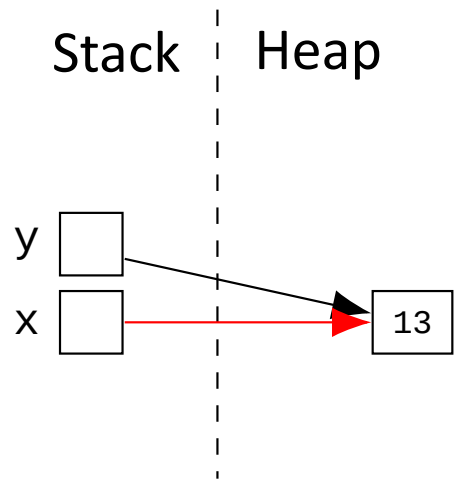
```
int* x { new int{13} };
int* y { x };
delete x;
delete y;
```



Dynamic Memory

Double delete

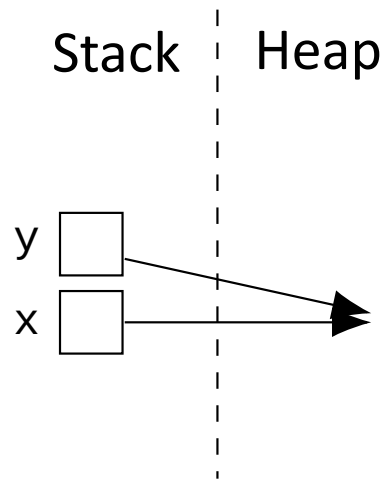
```
int* x { new int{13} };
int* y { x };
delete x;
delete y;
```



Dynamic Memory

Double delete

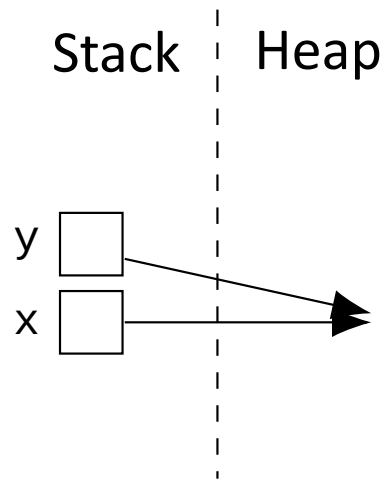
```
int* x { new int{13} };
int* y { x };
delete x;
delete y;
```



Dynamic Memory

Double delete

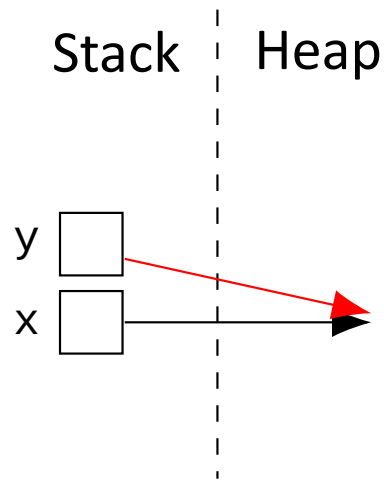
```
int* x { new int{13} };
int* y { x };
delete x;
delete y;
```



Dynamic Memory

Double delete

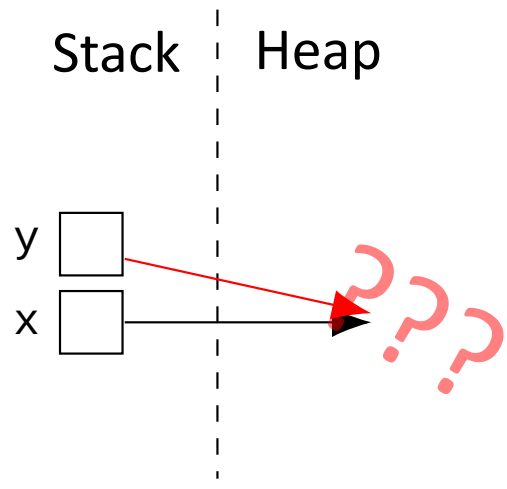
```
int* x { new int{13} };
int* y { x };
delete x;
delete y;
```



Dynamic Memory

Double delete

```
int* x { new int{13} };
int* y { x };
delete x;
delete y;
```



Dynamic Memory

The biggest issue you can run into

```
int* x;
cout << *x << endl;
```

Dynamic Memory

The biggest issue you can run into

```
int* x;
cout << *x << endl;
```

Segmentation fault

Dynamic Memory

The biggest issue you can run into

```
int* x { }; // x == nullptr
cout << *x << endl;
```

Dynamic Memory

The biggest issue you can run into

```
int* x { }; // x == nullptr
cout << *x << endl;
```

Segmentation fault

Dynamic Memory

The biggest issue you can run into

```
int* x { new int{5} };
// ...
delete x;
cout << *x << endl;
```

Dynamic Memory

The biggest issue you can run into

```
int* x { new int{5} };
// ...
delete x;
cout << *x << endl;
```

Segmentation fault

LINKÖPING UNIVERSITY

Structs & Pointers

An (faulty) example

```
struct Organization {
  std::string name;
};

struct Employee {
  std::string name;
  Organization employer;
};

void print(Employee const& e)
{
  cout << e.name << " works at "
       << e.employer.name << endl;
}
```

```
Organization liu {"LiU"};

Employee teacher {"Christoffer", liu};

Employee examiner {"Klas", liu};

// teacher and examiner works
// at LiU in the IDA department
liu.name = "LiU / IDA";

// What is printed?
print(teacher);
print(examiner);
```

Structs & Pointers

An (faulty) example

```
struct Organization {
  std::string name;
};

struct Employee {
  std::string name;
  Organization employer;
};

void print(Employee const& e)
{
  cout << e.name << " works at "
       << e.employer.name << endl;
}
```

```
Organization liu {"LiU"};

Employee teacher {"Christoffer", liu};

Employee examiner {"Klas", liu};

// teacher and examiner works
// at LiU in the IDA department
liu.name = "LiU / IDA";

// What is printed?
print(teacher);
print(examiner);
```

```
Christoffer works at LiU
Klas works at LiU
```

Structs & Pointers

An (faulty) example

```
struct Organization {
  std::string name;
};

struct Employee {
  std::string name;
  Organization employer;
};

void print(Employee const& e)
{
  cout << e.name << " works at "
       << e.employer.name << endl;
}
```

```
Organization liu {"LiU"};

Employee teacher {"Christoffer", liu};

Employee examiner {"Klas", liu};

// teacher and examiner works
// at LiU in the IDA department
liu.name = "LiU / IDA";

// What is printed?
print(teacher);
print(examiner);
```

```
Christoffer works at LiU
Klas works at LiU
```

Why?

Structs & Pointers

An (faulty) example

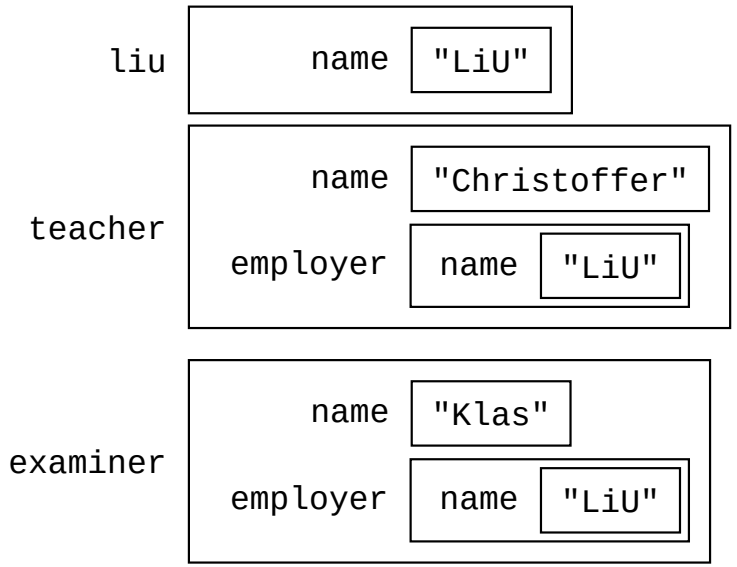
```
Organization liu {"LiU"};

Employee teacher {"Christoffer", liu};

Employee examiner {"Klas", liu};

// teacher and examiner works
// at LiU in the IDA department
liu.name = "LiU / IDA";

// What is printed?
print(teacher);
print(examiner);
```



Structs & Pointers

An (faulty) example

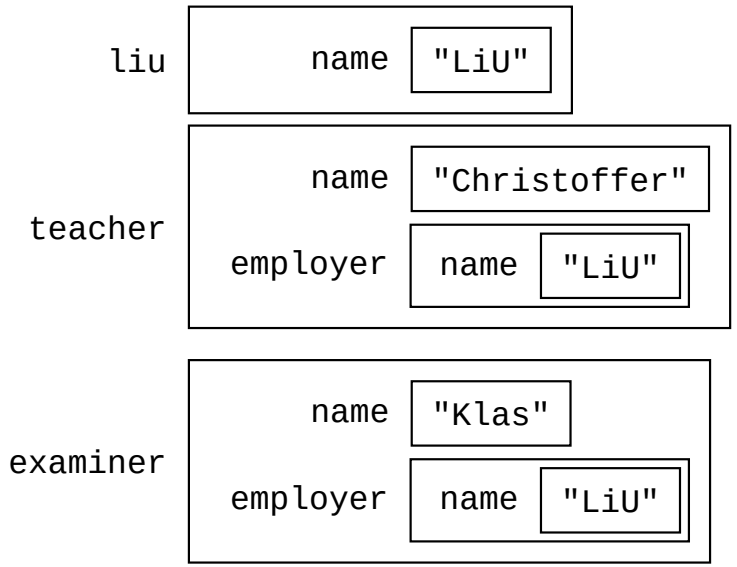
```
Organization liu {"LiU"};

Employee teacher {"Christoffer", liu};

Employee examiner {"Klas", liu};

// teacher and examiner works
// at LiU in the IDA department
liu.name = "LiU / IDA";

// What is printed?
print(teacher);
print(examiner);
```



Structs & Pointers

An (faulty) example

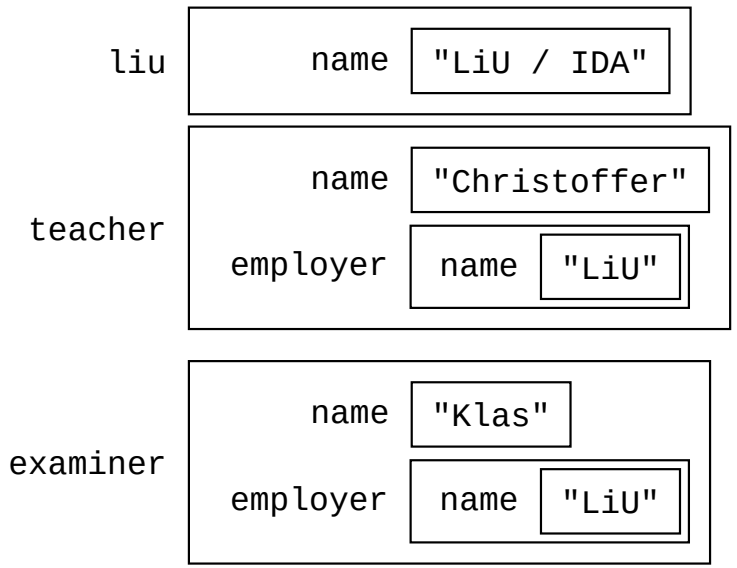
```
Organization liu {"LiU"};

Employee teacher {"Christoffer", liu};

Employee examiner {"Klas", liu};

// teacher and examiner works
// at LiU in the IDA department
liu.name = "LiU / IDA";

// What is printed?
print(teacher);
print(examiner);
```



Structs & Pointers

A better implementation!

```
struct Organization {
  std::string name;
};

struct Employee {
  std::string name;
  Organization* employer;
};

void print(Employee const& e)
{
  cout << e.name << " works at "
       << e.employer->name << endl;
}
```

```
Organization liu {"LiU"};

Employee teacher {"Christoffer", &liu};

Employee examiner {"Klas", &liu};

// teacher and examiner works
// at LiU in the IDA department
liu.name = "LiU / IDA";

// What is printed?
print(teacher);
print(examiner);
```

Structs & Pointers

A better implementation!

```
struct Organization {
  std::string name;
};

struct Employee {
  std::string name;
  Organization* employer;
};

void print(Employee const& e)
{
  cout << e.name << " works at "
       << e.employer->name << endl;
}
```

```
Organization liu {"LiU"};

Employee teacher {"Christoffer", &liu};

Employee examiner {"Klas", &liu};

// teacher and examiner works
// at LiU in the IDA department
liu.name = "LiU / IDA";

// What is printed?
print(teacher);
print(examiner);
```

Structs & Pointers

A better implementation!

```
struct Organization {
  std::string name;
};

struct Employee {
  std::string name;
  Organization* employer;
};

void print(Employee const& e)
{
  cout << e.name << " works at "
       << e.employer->name << endl;
}
```

```
Organization liu {"LiU"};

Employee teacher {"Christoffer", &liu};

Employee examiner {"Klas", &liu};

// teacher and examiner works
// at LiU in the IDA department
liu.name = "LiU / IDA";

// What is printed?
print(teacher);
print(examiner);
```

```
Christoffer works at LiU / IDA
Klas works at LiU / IDA
```

Structs & Pointers

A better implementation!

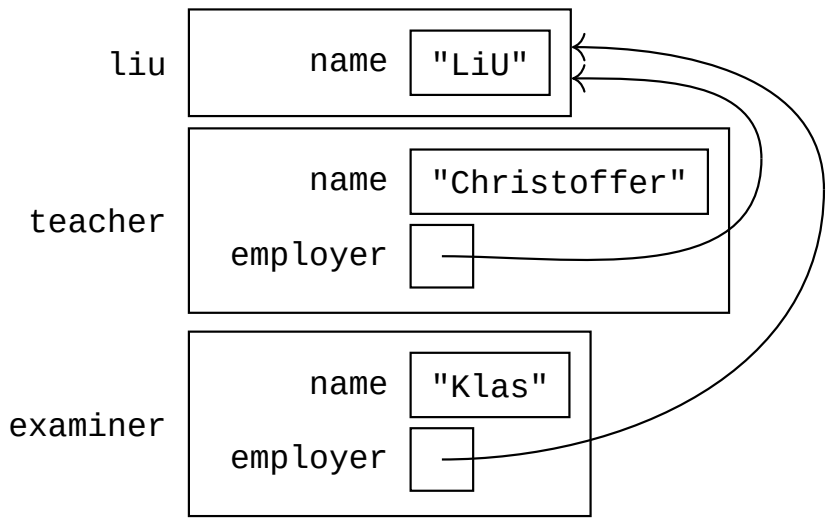
```
Organization liu {"LiU"};

Employee teacher {"Christoffer", &liu};

Employee examiner {"Klas", &liu};

// teacher and examiner works
// at LiU in the IDA department
liu.name = "LiU / IDA";

// What is printed?
print(teacher);
print(examiner);
```



Structs & Pointers

A better implementation!

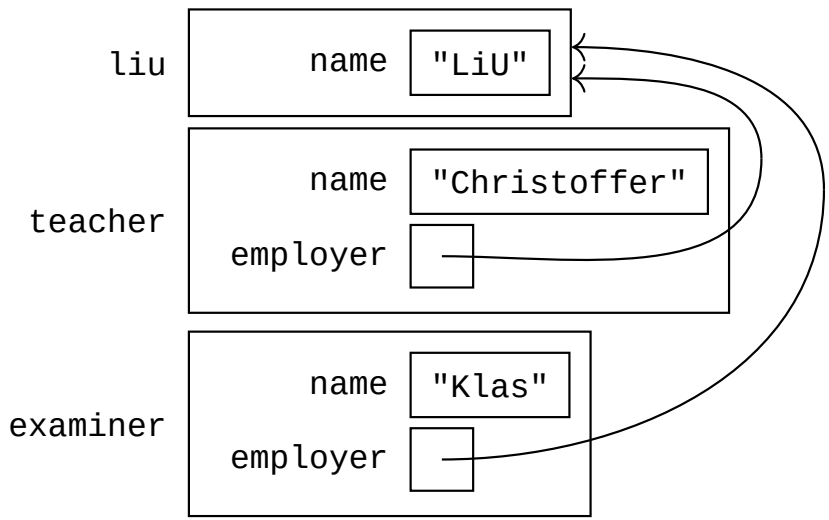
```
Organization liu {"LiU"};

Employee teacher {"Christoffer", &liu};

Employee examiner {"Klas", &liu};

// teacher and examiner works
// at LiU in the IDA department
liu.name = "LiU / IDA";

// What is printed?
print(teacher);
print(examiner);
```



Structs & Pointers

A better implementation!

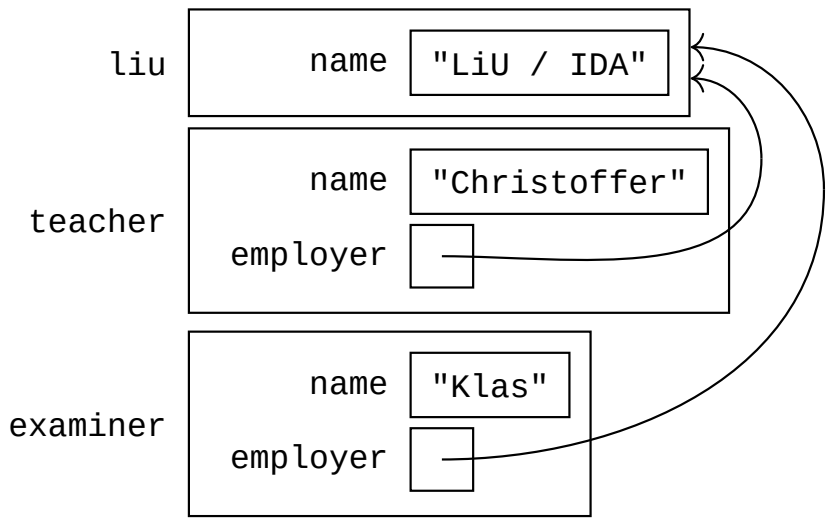
```
Organization liu {"LiU"};

Employee teacher {"Christoffer", &liu};

Employee examiner {"Klas", &liu};

// teacher and examiner works
// at LiU in the IDA department
liu.name = "LiU / IDA";

// What is printed?
print(teacher);
print(examiner);
```



www.liu.se

LINKÖPING UNIVERSITY