Fundamentals of Computing: Lecture 14

Piyush P Kurur
Office no: 224
Dept. of Comp. Sci. and Engg.
IIT Kanpur

August 31, 2009

Pointers

Pointers

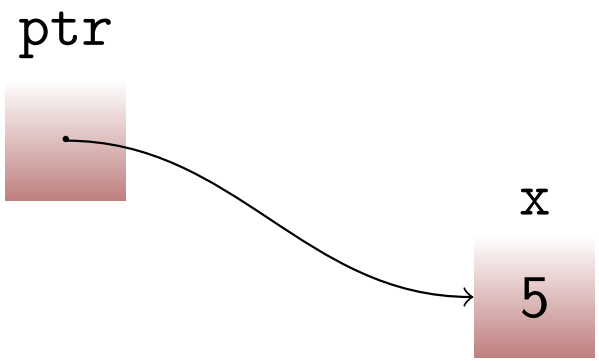
- A pointer is an abstraction of memory address.

Pointers

- A pointer is an abstraction of memory address.
- Value of a pointer variable of type `T` is an address of a memory cell capable of storing a value of type `T`.

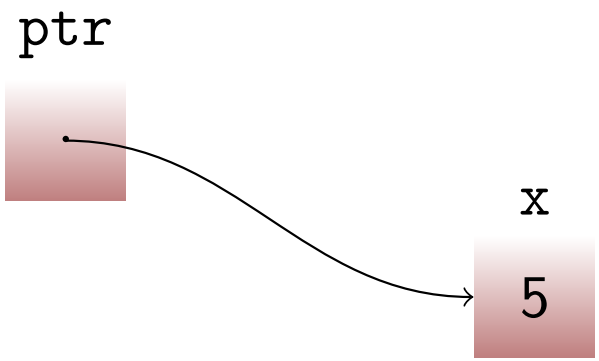
Pointers

- A pointer is an abstraction of memory address.
- Value of a pointer variable of type `T` is an address of a memory cell capable of storing a value of type `T`.



Pointers

- A pointer is an abstraction of memory address.
- Value of a pointer variable of type `T` is an address of a memory cell capable of storing a value of type `T`.



Before we start:

WARNING

Too much of pointer gymnastics can cause serious injury to readability.

Declaration

Let `T` be a type then we can declare a pointer to `T` as `T *ptr`

Declaration

Let `T` be a type then we can declare a pointer to `T` as `T *ptr`

Example

```
int x, *p, **pp;
```

Declaration

Let `T` be a type then we can declare a pointer to `T` as `T *ptr`

Example

```
int x, *p, **pp;
```

The above code declares

- `x` is an integer
- `p` is a pointer to an integer
- `pp` is a pointer to a pointer to an integer.

Dereferencing

For a pointer variable `ptr`, `*ptr` is the value stored in the location pointed by `ptr`.

Dereferencing

For a pointer variable `ptr`, `*ptr` is the value stored in the location pointed by `ptr`.

Conversely

If `x` is a variable then `&x` is the address of the variable.

Dereferencing

For a pointer variable `ptr`, `*ptr` is the value stored in the location pointed by `ptr`.

Conversely

If `x` is a variable then `&x` is the address of the variable.

```
int x, *p, *pp;
```

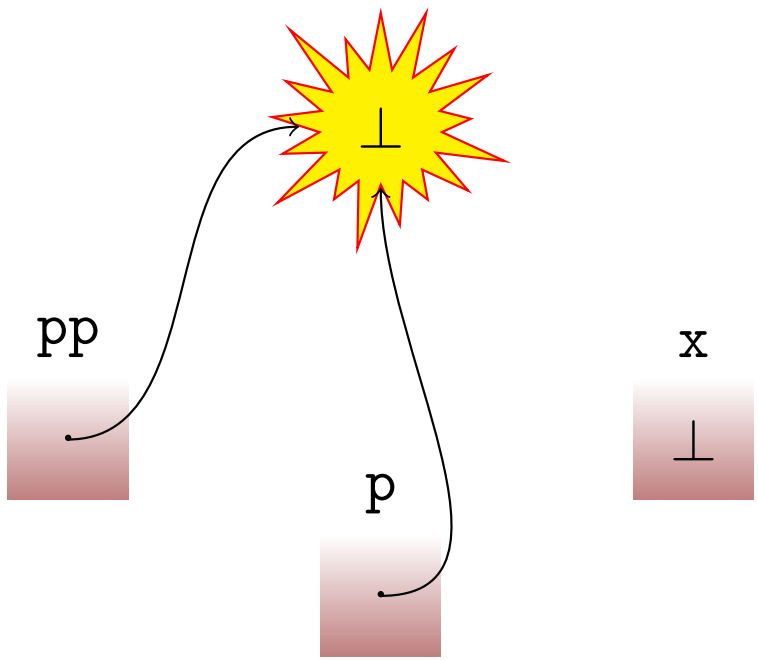
Dereferencing

For a pointer variable `ptr`, `*ptr` is the value stored in the location pointed by `ptr`.

Conversely

If `x` is a variable then `&x` is the address of the variable.

```
int x, *p, *pp;
```



Dereferencing

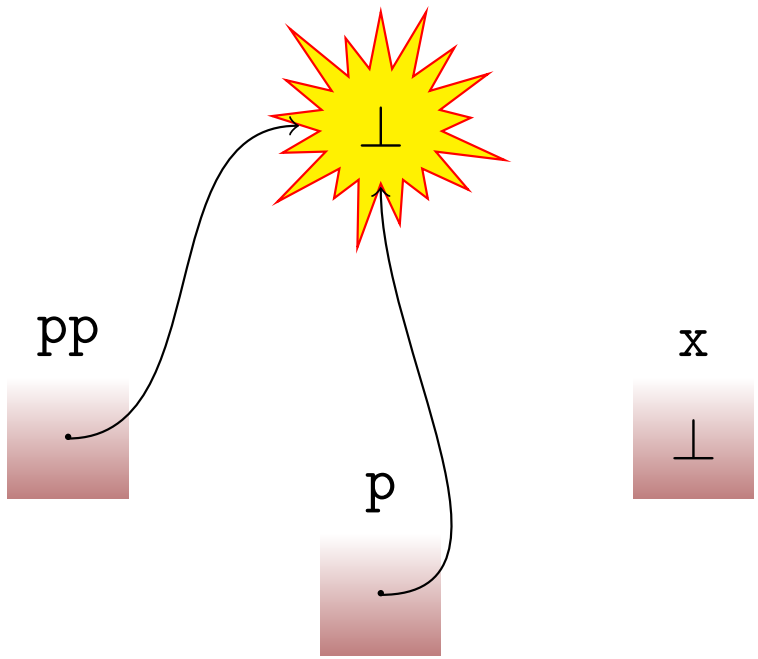
For a pointer variable `ptr`, `*ptr` is the value stored in the location pointed by `ptr`.

Conversely

If `x` is a variable then `&x` is the address of the variable.

```
int x, *p, *pp;

x =5;
p = &x;
pp = &p;
```



Dereferencing

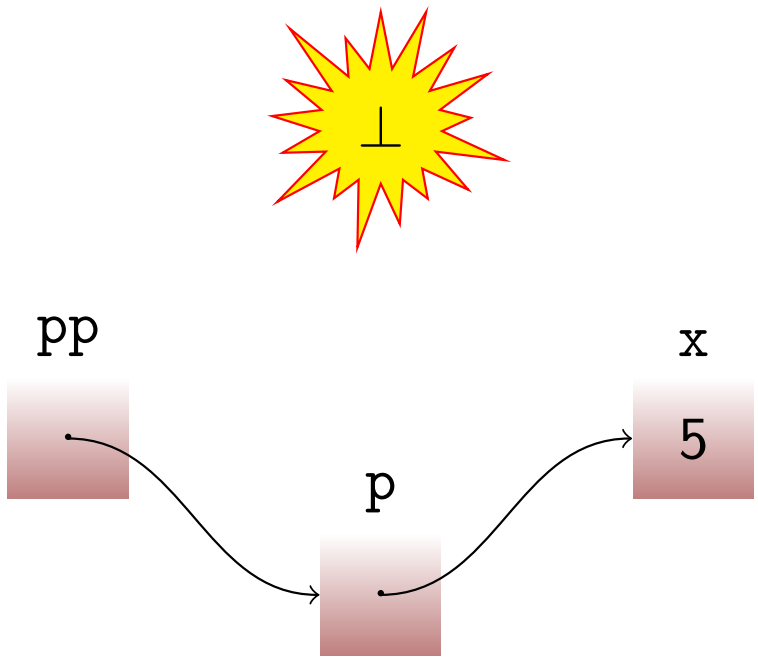
For a pointer variable `ptr`, `*ptr` is the value stored in the location pointed by `ptr`.

Conversely

If `x` is a variable then `&x` is the address of the variable.

```
int x, *p, *pp;

x =5;
p = &x;
pp = &p;
```

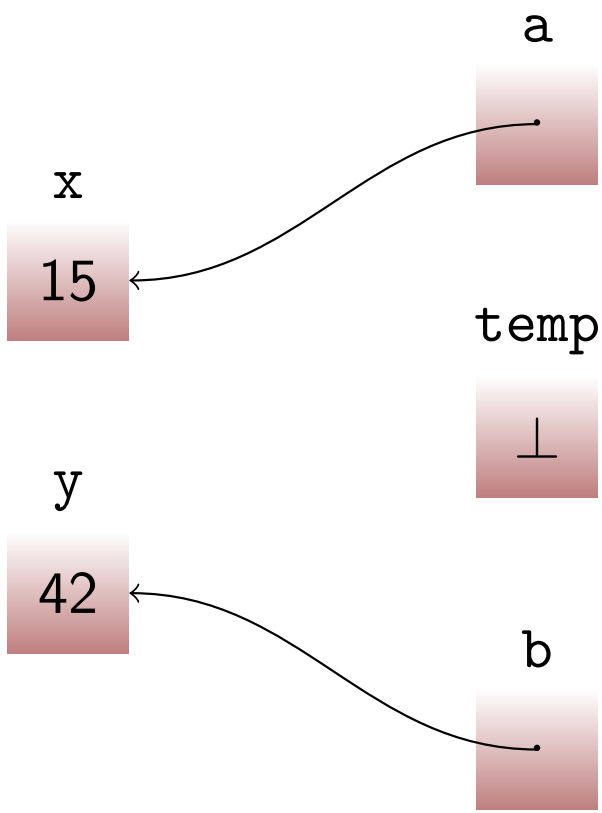


Swapping: Pointer version

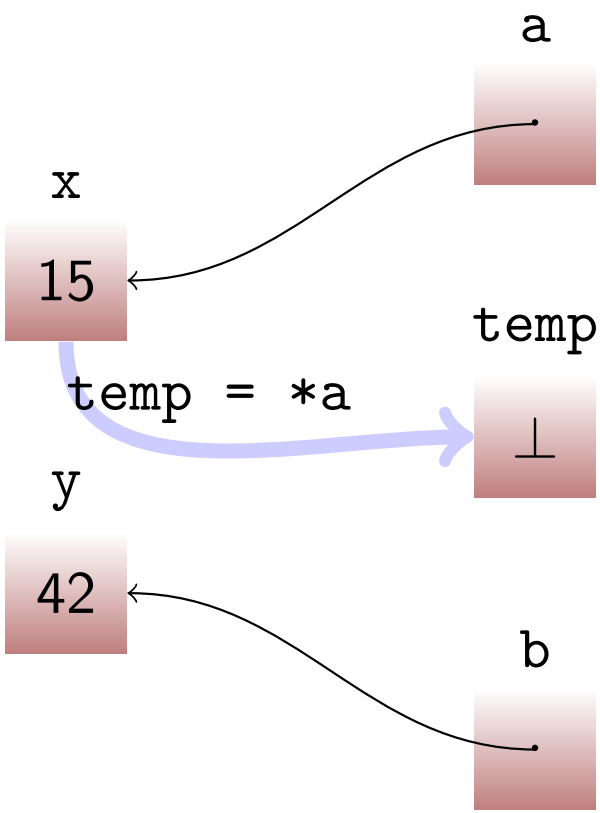
```
void swap(int *, int *);
int main()
{
  int x=15,y=42;
  printf("x = %d, y = %d\n",x,y);
  swap(&x,&y);
  printf("x = %d, y = %d\n",x,y);
}
void swap(int *a, int *b)
{
  int temp;
  temp = *a;
  *a = *b;
  *b = temp;
}
```

Swapping Pointer version

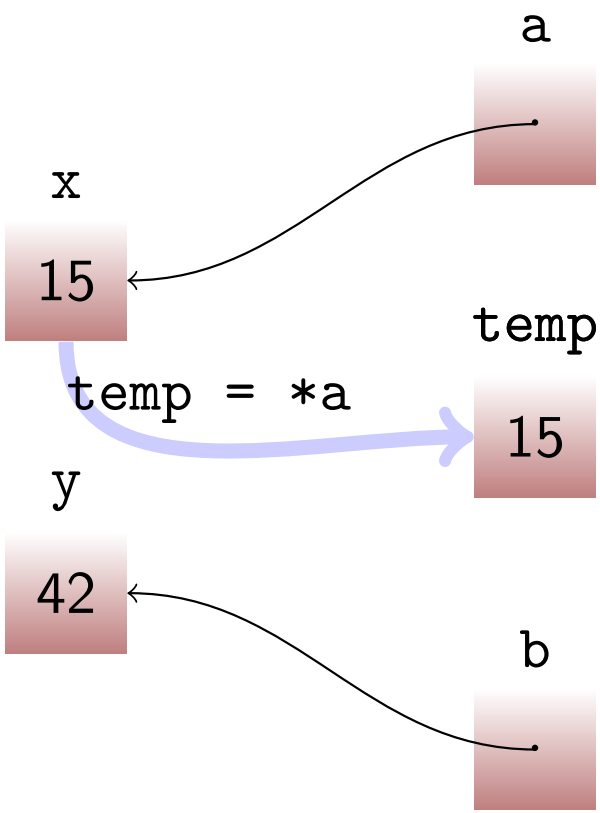
Swapping Pointer version



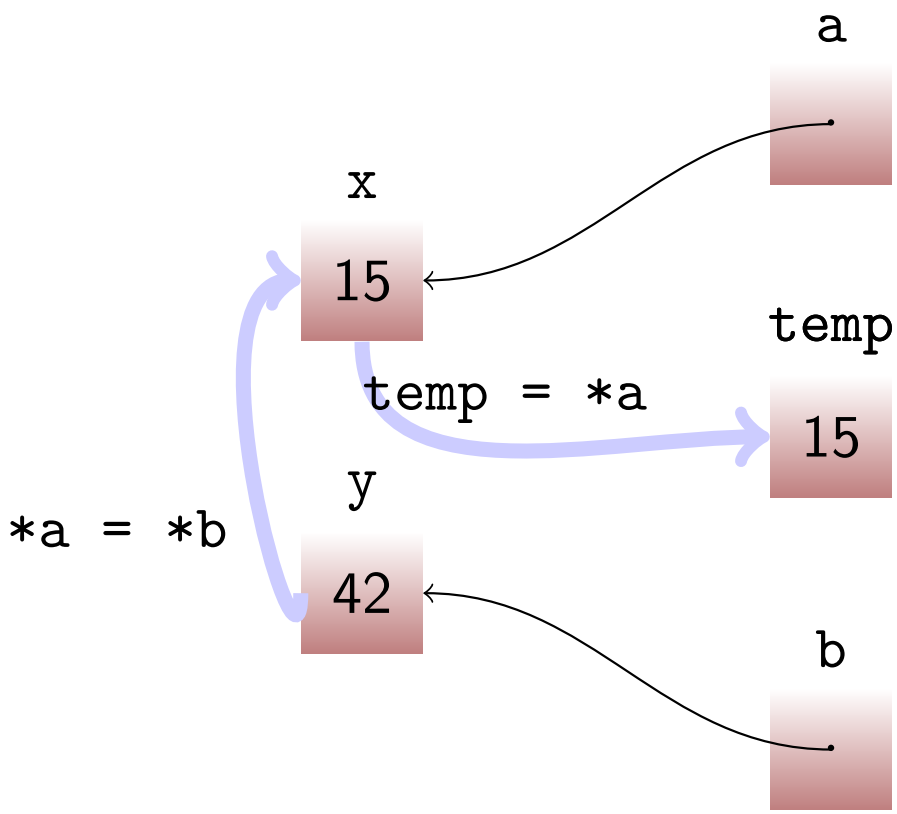
Swapping Pointer version



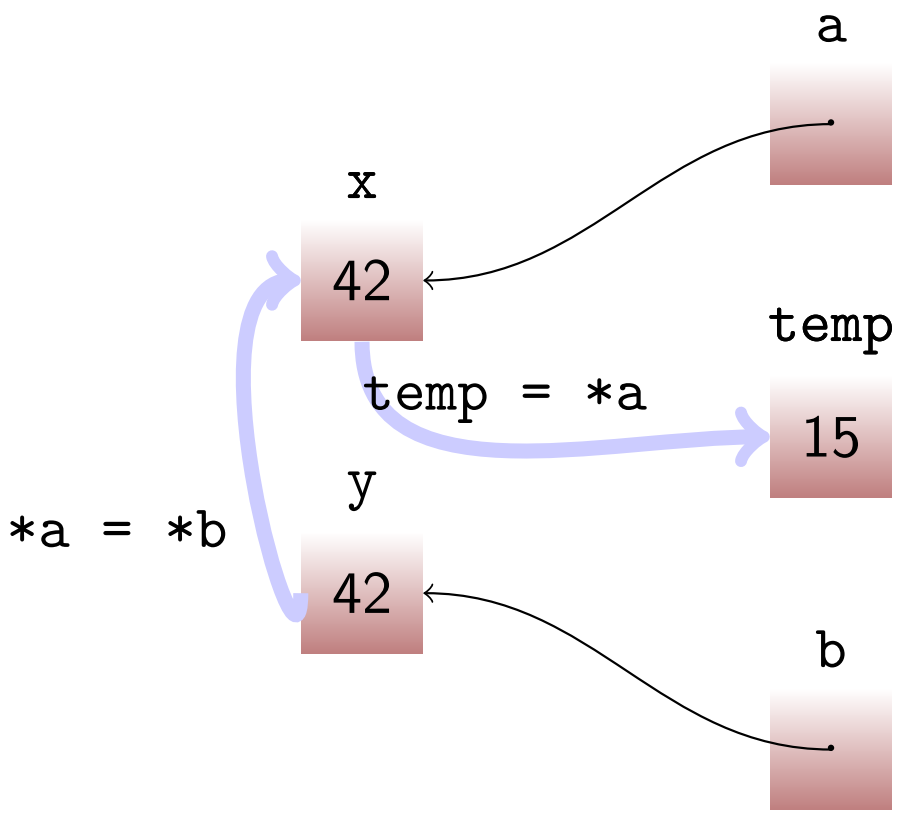
Swapping Pointer version



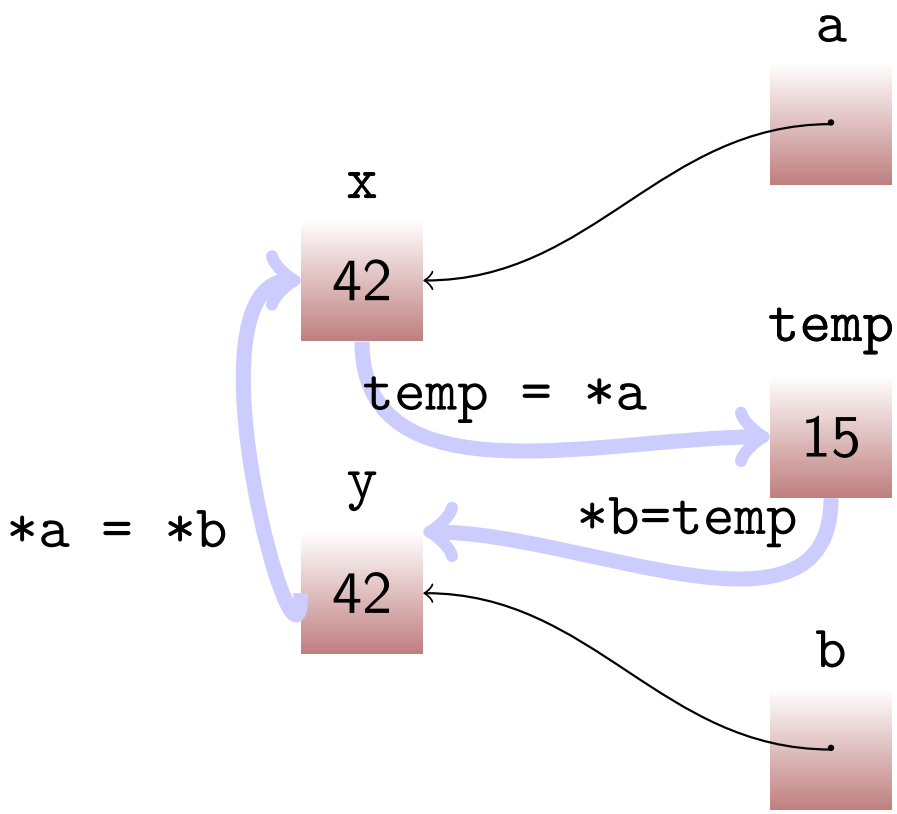
Swapping Pointer version



Swapping Pointer version



Swapping Pointer version



Swapping Pointer version