Fundamentals of Computing: Lecture 15

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

September 2, 2009

Summary of the previous lecture

Summary of the previous lecture

- A pointer is an abstraction of memory address.

Summary of the previous lecture

- A pointer is an abstraction of memory address.
- To declare use `T *ptr;`

Summary of the previous lecture

- A pointer is an abstraction of memory address.
- To declare use `T *ptr;`
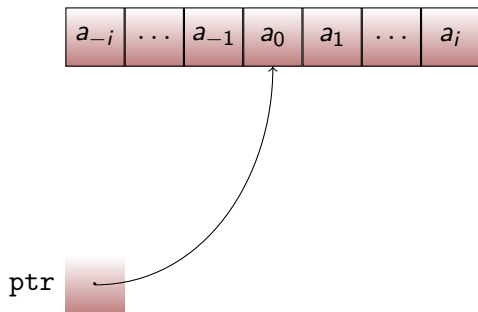- A pointer to type `T` has as value an address of a memory cell capable of storing a value of type `T`.

Summary of the previous lecture

- A pointer is an abstraction of memory address.
- To declare use T *ptr;
- A pointer to type T has as value an address of a memory cell capable of storing a value of type T.
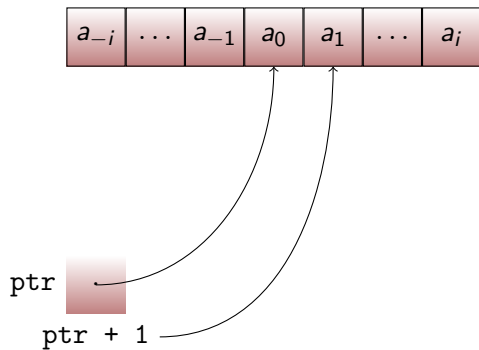- The expression *ptr is the value stored at the location pointed by ptr. *ptr is also an l-value and can be assigned.

Summary of the previous lecture

- A pointer is an abstraction of memory address.
- To declare use T *ptr;
- A pointer to type T has as value an address of a memory cell capable of storing a value of type T.
- The expression *ptr is the value stored at the location pointed by ptr. *ptr is also an l-value and can be assigned.
- For a variable x, &x is the address of the variable. The expression &x is *not* an l-value.

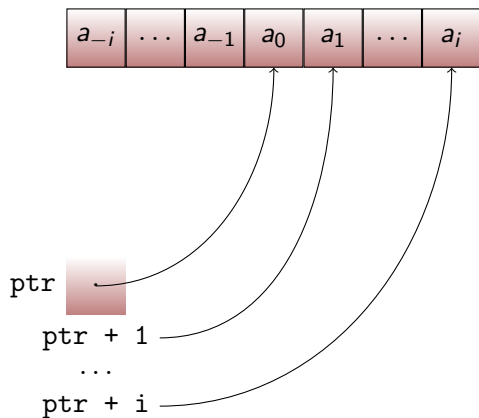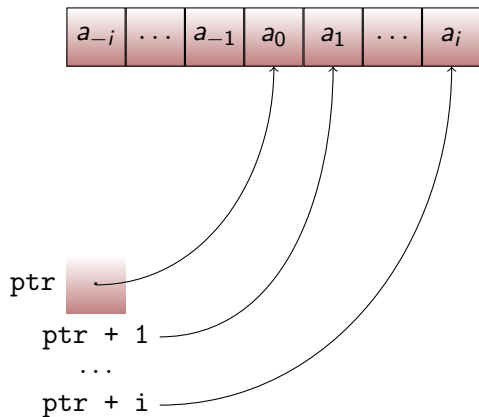# Pointer arithmetic

| a_{-i} | $\ldots$ | a_{-1} | a_0 | a_1 | $\ldots$ | a_i |

ptr

Pointer arithmetic

Pointer arithmetic

The diagram shows an array of cells labeled (left to right): a_{-i}, $\ldots$, a_{-1}, a_0, a_1, $\ldots$, a_i.

Below the array:

```
ptr
  ptr + 1
    ...
  ptr + i
```

`ptr` points to a_0, `ptr + 1` points to a_1, and `ptr + i` points to a_i.

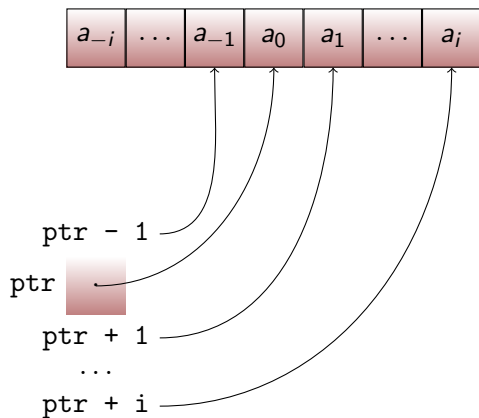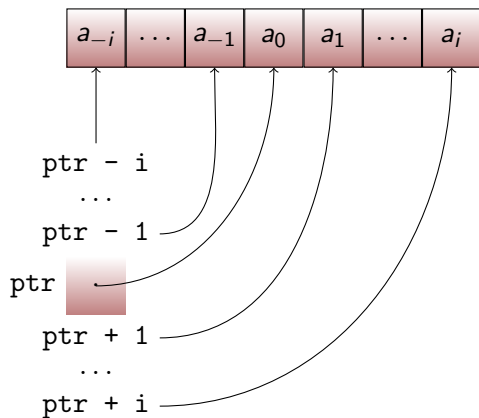# Pointer arithmetic

Pointer arithmetic

Pointer arithmetic

What operations are allowed on pointers?
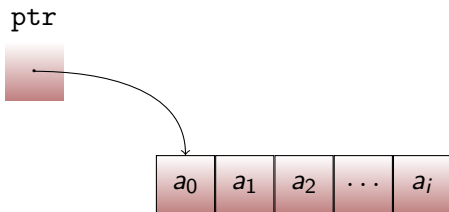
- One can add any integer to a pointer of any type `ptr + i` is the pointer to the ith location starting from `ptr`.
- One can subtracte any integer to a pointer of any type
- If `ptr1` and `ptr2` are pointers to the same type then `ptr1 - ptr2`. It is the number of cells between `ptr1` and `ptr2`.
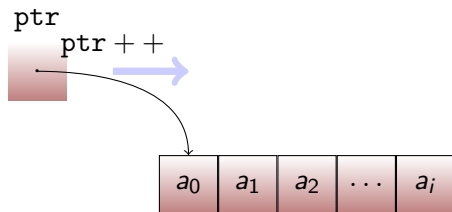- `ptr++`, `ptr--` etc makes sense because `ptr +1` and `ptr -1` makes sense

`ptr`

What operations are allowed on pointers?

- One can add any integer to a pointer of any type `ptr + i` is the pointer to the ith location starting from `ptr`.
- One can subtracte any integer to a pointer of any type
- If `ptr1` and `ptr2` are pointers to the same type then `ptr1 - ptr2`. It is the number of cells between `ptr1` and `ptr2`.
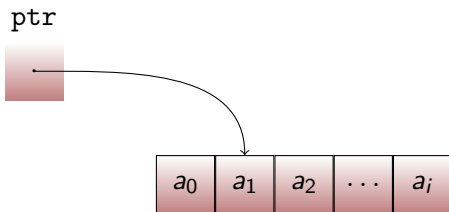- `ptr++`, `ptr--` etc makes sense because `ptr +1` and `ptr -1` makes sense

ptr

What operations are allowed on pointers?

- One can add any integer to a pointer of any type `ptr + i` is the pointer to the ith location starting from `ptr`.
- One can subtracte any integer to a pointer of any type
- If `ptr1` and `ptr2` are pointers to the same type then `ptr1 - ptr2`. It is the number of cells between `ptr1` and `ptr2`.
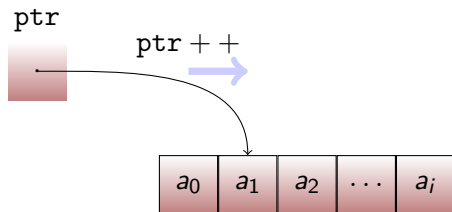- `ptr++`, `ptr--` etc makes sense because `ptr +1` and `ptr -1` makes sense

What operations are allowed on pointers?

- One can add any integer to a pointer of any type `ptr + i` is the pointer to the ith location starting from `ptr`.
- One can subtracte any integer to a pointer of any type
- If `ptr1` and `ptr2` are pointers to the same type then `ptr1 - ptr2`. It is the number of cells between `ptr1` and `ptr2`.
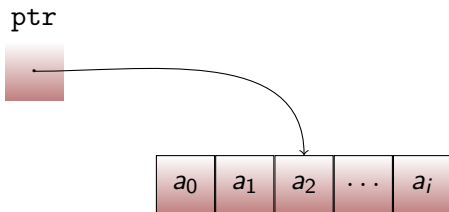- `ptr++`, `ptr--` etc makes sense because `ptr +1` and `ptr -1` makes sense

ptr

What operations are allowed on pointers?

- One can add any integer to a pointer of any type `ptr + i` is the pointer to the ith location starting from `ptr`.
- One can subtracte any integer to a pointer of any type
- If `ptr1` and `ptr2` are pointers to the same type then `ptr1 - ptr2`. It is the number of cells between `ptr1` and `ptr2`.
- `ptr++`, `ptr--` etc makes sense because `ptr +1` and `ptr -1` makes sense

What operations are allowed on pointers?

- One can add any integer to a pointer of any type `ptr + i` is the pointer to the ith location starting from `ptr`.
- One can subtracte any integer to a pointer of any type
- If `ptr1` and `ptr2` are pointers to the same type then `ptr1 - ptr2`. It is the number of cells between `ptr1` and `ptr2`.
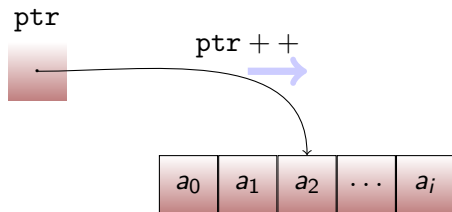- `ptr++`, `ptr--` etc makes sense because `ptr +1` and `ptr -1` makes sense

ptr

What operations are allowed on pointers?

- One can add any integer to a pointer of any type `ptr + i` is the pointer to the ith location starting from `ptr`.
- One can subtracte any integer to a pointer of any type
- If `ptr1` and `ptr2` are pointers to the same type then `ptr1 - ptr2`. It is the number of cells between `ptr1` and `ptr2`.
- `ptr++`, `ptr--` etc makes sense because `ptr +1` and `ptr -1` makes sense

ptr

$ptr + +$

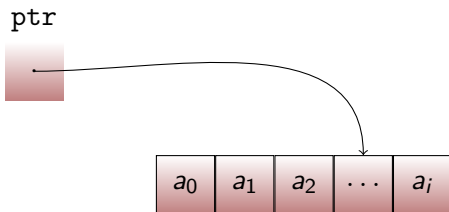# What operations are allowed on pointers?

- One can add any integer to a pointer of any type `ptr + i` is the pointer to the ith location starting from `ptr`.
- One can subtracte any integer to a pointer of any type
- If `ptr1` and `ptr2` are pointers to the same type then `ptr1 - ptr2`. It is the number of cells between `ptr1` and `ptr2`.
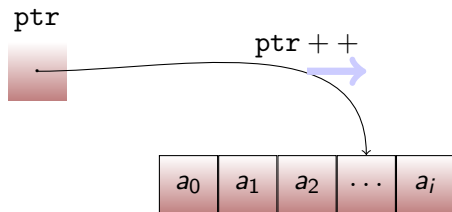- `ptr++`, `ptr--` etc makes sense because `ptr +1` and `ptr -1` makes sense

ptr

What operations are allowed on pointers?

- One can add any integer to a pointer of any type `ptr + i` is the pointer to the ith location starting from `ptr`.
- One can subtracte any integer to a pointer of any type
- If `ptr1` and `ptr2` are pointers to the same type then `ptr1 - ptr2`. It is the number of cells between `ptr1` and `ptr2`.
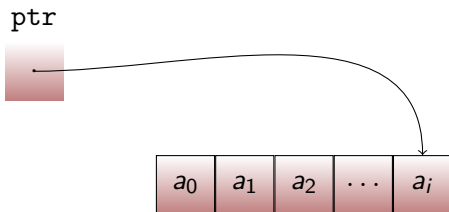- `ptr++`, `ptr--` etc makes sense because `ptr +1` and `ptr -1` makes sense

What operations are allowed on pointers?

▶ One can add any integer to a pointer of any type `ptr + i` is the pointer to the ith location starting from `ptr`.

▶ One can subtracte any integer to a pointer of any type

▶ If `ptr1` and `ptr2` are pointers to the same type then `ptr1 - ptr2`. It is the number of cells between `ptr1` and `ptr2`.

▶ `ptr++`, `ptr--` etc makes sense because `ptr +1` and `ptr -1` makes sense

Relation with arrays

Consider

```
int a[100];
```

Relation with arrays

Consider

```
int a[100];
```

- The value a is the address of the first element.

Relation with arrays

Consider

```
int a[100];
```

- The value a is the address of the first element.
- The value a + i is equivalent to &a[i]

Relation with arrays

Consider

```
int a[100];
```

- The value a is the address of the first element.
- The value a + i is equivalent to &a[i]
- More interestingly a[i] is same as *(a+i)

Relation with arrays

Consider

```
int a[100];
```

- The value a is the address of the first element.
- The value a + i is equivalent to &a[i]
- More interestingly a[i] is same as *(a+i)

```
int a[100], *ptr;
for(ptr=a; ptr - a < 100; ptr++)
{
  *ptr = 0;
}
```