



# SLIIT

*Discover Your Future*

## IT1050- Object Oriented Concepts

### Pointers in C++



SLIIT  
FACULTY OF COMPUTING

# Introduction

- A *pointer* is a variable that **holds a memory address**.
- Why pointers?
  - Provide the means by which functions can modify the arguments of the calling function.  
*Note: pass by reference was not available in C.*
  - Pointers support dynamic allocation of memory.
  - Can improve the efficiency of some routines.

# *Declaring a Pointer*

Syntax:

`type *name;`

Examples:

`int *m;`

`char *array;`

`double *temp;`

`int **matrix;`

Memory	
Address	Contents
0x3267A1B0	0x32
0x3267A1B1	67
0x3267A1B2	A1
0x3267A1B3	B8
0x3267A1B4	
0x3267A1B5	
0x3267A1B6	
0x3267A1B7	
0x3267A1B8	
0x3267A1B9	
0x3267A1BA	
0x3267A1BB	

m



# *Pointer Operators*

- Once a pointer is declared, the operator **\*** can be used to obtain the value located at the address that is held by the pointer.
  - Eg : **cout**<< **\*m** ; // will print the value at the location pointed by m.
- The operator **&** can be used to obtain the memory address of an operand.
  - Eg : **cout** << **&a**; // will print the memory address of a.
- Both the operators, **\*** and **&** are unary operators. That is, it uses only one operand.

# Pointer Operators

## Declaring a pointer

- Example:

```
int *m;
```

Note :

Initially the pointer will point at  
( will store the memory address  
of ) any location in the memory.

Memory	
Address	Contents
<b>0x3267A1B0</b>	0x32
<b>0x3267A1B1</b>	67
<b>0x3267A1B2</b>	A1
<b>0x3267A1B3</b>	B8
<b>0x3267A1B4</b>	
<b>0x3267A1B5</b>	
<b>0x3267A1B6</b>	
<b>0x3267A1B7</b>	
<b>0x3267A1B8</b>	
<b>0x3267A1B9</b>	
<b>0x3267A1BA</b>	
<b>0x3267A1BB</b>	

# Pointer Operators

- Example:

```
int *m;
```

```
int count = 7;
```

```
// count is a variable that stores an  
integer.
```

Memory	
Address	Contents
m 0x3267A1B0	0x32
0x3267A1B1	67
0x3267A1B2	A1
0x3267A1B3	B8
count 0x3267A1B4	0
0x3267A1B5	0
0x3267A1B6	0
0x3267A1B7	7
0x3267A1B8	
0x3267A1B9	
0x3267A1BA	
0x3267A1BB	



# Pointer Operators

- Example:

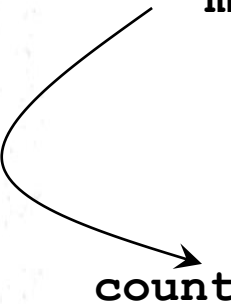
```
int *m;
```

```
int count = 7;
```

```
m = &count;
```

```
// Assigns the memory address of  
count to the pointer m
```

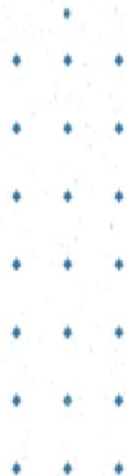
- The value of **m** is 0x3267A1B4
- The value of **\*m** is 7



Memory	
Address	Contents
0x3267A1B0	0x32
0x3267A1B1	67
0x3267A1B2	A1
0x3267A1B3	B4
0x3267A1B4	0
0x3267A1B5	0
0x3267A1B6	0
0x3267A1B7	7
0x3267A1B8	
0x3267A1B9	
0x3267A1BA	
0x3267A1BB	

# *Dynamic Memory Allocation*

- Allows a program to obtain memory at runtime.
- Dynamic Allocation Functions in C++
  - **new** is used to allocate memory.
  - **delete** is used to release memory.





# C++

- Allocate Memory for an integer variable

```
int * p ;  
p = new int;
```

```
// int *p = new int;
```

- Deallocate memory

```
delete p;
```

# *Variable Declarations*

//Declaring an integer Variable and initializing to 87

```
int *p;
```

```
p = new int (87); //Initializes to 87. That is *p is 87
```

```
if (!p){ // check whether memory was allocated
```

```
    cout << "Cannot Allocate Memory" << endl;
```

```
    exit(1);
```

```
}
```

```
delete p;
```

# *Array Declarations*

//Declaring an integer array of 10 elements

```
int *k;
```

```
k = new int [10]; //an array of 10 integers
```

```
if (!k) {
```

```
    cout << "Cannot Allocate Memory" << endl;
```

```
    exit(1);
```

```
}
```

```
delete [] k;
```

# *Object Declarations*

//Declaring an object of Rectangle

```
Rectangle *r;
```

```
r = new int Rectangle();
```

```
if (!r) {
```

```
    cout << "Cannot Allocate Memory" << endl;
```

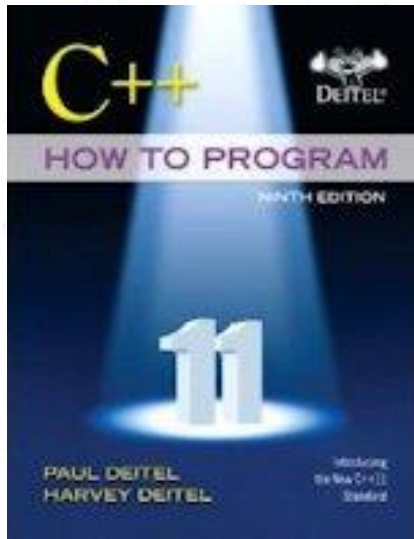
```
    exit(1);
```

```
}
```

```
Delete r;
```

```
Rectangle *r = new Rectangle();
```

# Reference



## Chapter 08

Deitel & Deitel's (2016), C++ How to Program,  
9<sup>th</sup> Edition