Pointers

Mohammad Tasin

Agenda

- Pointer
- Referencing operators
- Dereferencing operators
- Wild pointer
- NULL pointer
- Dangling pointer
- Void pointer
- this pointer

Pointer

- Pointer is a variable, which contains address of another variable
- Pointer is declared using * symbol
- Pointer content is considered address of the variable of type same as the type of the pointer
- Size of pointer is fixed and depends an OS and computer architecture

Referencing operators

int x = 10;

&

- **Address of operator**
- **Referencing operator**
- **Unary operator**
- **&____**

Variable only

- **This operator take variable and return Address**
- **We are print any variable Address**

- **Syntax ⟶ cout<<&x;**

Dereferencing operators

*

- Indirection operator
- Dereferencing operator
- Unary operator

- *

Address only

- This operator take **Address** and return **Variable**

Syntax → *&x == x

- int x = 10;
- &x = 20; → Error
- char str[20];
- str = “TasiNCoder”; → Error
- &x is not a variable it is just a way to represent address of variable. Which is considered as a constant value. We Known that in the left hand side of assignment operator, a variable must be present.

Wild pointer

- An uninitialized pointer is a wild pointer

```
void F1()
{
    int *p;
    *p = 10;
    ........;
    ........;
}
```

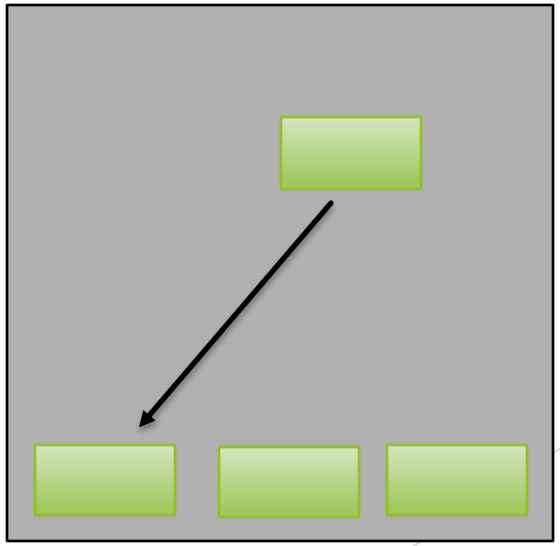
int *p; → wild pointer

*p = 10; → illegal use of pointer

NULL Pointer

- A pointer containing NULL (special address) is known as NULL pointer
- If a pointer containing NULL, we consider it as if it is not pointing to any location
- As a safeguard to illegal use of pointers you can check for NULL before accessing pointer variable

```
void F1()
{
    int *p = NULL;
    ----
    ----
    if(p!=NULL)
        *p = 10;
}
```



Dangling pointer

- A pointer pointing to a memory location that has been deleted is called dangling pointer.

```
int* F1()
{
    int a;
    return &a;
}
void F2()
{
    int *p;
    p = F1();
    *p = 10;
}
```

p is a dangling Pointer

illegal use of pointer

Void pointer

- void pointer is a generic pointer that has no associated data type with it.
- void pointer can hold address of any type.

```
void *p;
int x;
p = &x;
```

```
void *p;
float z;
p = &z;
```

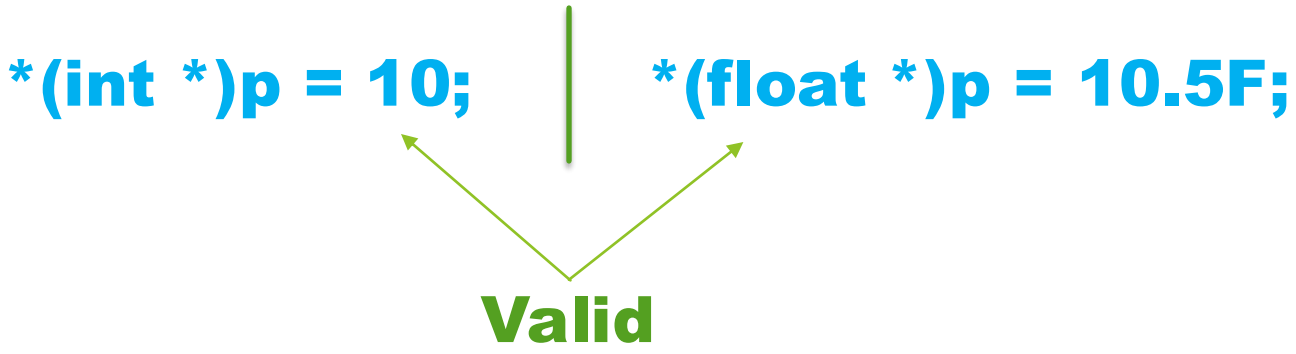
- void pointers cannot be dereferenced

```
*p = 10;
```

```
*p = 10.5F;
```

Error

- **However void pointer can be dereferenced using typecasting**



this pointer

- **this is a keyword.**
- **this is a local object pointer in every instance member function which contains address of the current object.**
- **this cannot be modified (you cannot change the value in the pointer).**

When to use this pointer

- **Name conflict between instance member variables and local variables.**
- **Whenever it is required to represent current object in instance member function.**