```
______object
         edject to mirror
 peration == "MIRROR.
 Trror mod.use_y = True
Trror_mod.use_y = False
"Irror_mod.use_z = False
 _operation == "MIRROR_Y"
"Irror_mod.use_x = False
mirror_mod.use_y = True
mirror_mod.use_z = False
  operation == "MIRROR Z"
  rror_mod.use_x = False
  lrror_mod.use_y = False
  rror_mod.use_z = True
  election at the end -add
  ob.select= 1
  er ob.select=1
   "Selected" + str Sdie SSION 4
bpy.context.selected w SSION 4
   ata.objects[one.name].sel
  int("please select ex
  operator classes - Mohamed Saied
   ypes.Operator):

X mirror to the selected
   ject.mir.ror_mirror_x"
   FOR X'
```

s is not

```
peration == "MIRROR_X":
                  mirror_mod.use_x = True
                  "Irror_mod.use_y = False
                  "Irror_mod.use_z = False
                    operation = "MIRROR_Y"
                  lrror_mod.use_x = False
                  lrror_mod.use_y = True
                  mirror_mod.use_z = False
                    operation == "MIRROR Z"
                    rror mod.use x = False
                    rror mod.use_y = False
                    rror_mod.use_z = True
                     ob.select= 1
Agenda
                     er ob.select=1
                     ntext.scene.objects.activ
                     "Selected" + str(modified
                     rror ob.select = 0
                      bpy.context.selected_ob
                     ta.objects[one.name].se
                     int("please select exact
                       OPERATOR CLASSES -
                     ypes.Operator):
                     X mirror to the select
                    ject.mirror_mirror_x"
                    POP Y
```

- Type casting
- iterator
- Pointer to function
- Lambda

is not

Implicit Conversion

• Done by the compiler on its own, without any external trigger from the user.

4 not

- Generally takes place when in an expression more than one data type is present. In such
 condition type conversion (type promotion) takes place to avoid lose of data.
- All the data types of the variables are upgraded to the data type of the variable with largest data type.
- bool -> char -> short int -> int ->
- unsigned int -> long -> unsigned ->
- long long -> float -> double -> long double

Int("please select exact!

eration == "MIRROR_X": Implcit conversion

_____or__mod_mirror_object

Ject to micro

```
// An example of implicit conversion Transport
                    rror_mod.use_y = False
#include <iostream>
                    rror_mod.use z = True
using namespace std;
                     election at the end -add
                     ob.select= 1
int main()
                     er_ob.select=1
                     ntext.scene.objects.acti
   int x = 10; // integer x
    char y = 'a'; // character celect = 0
                      bpy.context.selected_ob
    // y implicitly converted to int. ASCII
    // value of 'a' is 97
                     int("please select exact)
    X = X + Y;
                       OPERATOR CLASSES
    // x is implicitly converted to float
    float z = x + 1.0;
    cout << "x = " << x << endl
        << "y = " << y << end1 to the select
        << "z = " << z << endl;
    return 0;
                                     to not
```

Output: x = 107 y = a z = 108

Explicit conversion

- Explicit Type Conversion: This process is also called type casting and it is user-defined. Here the user can typecast the result to make it of a particular data type.
- In C++, it can be done by two ways:
- Converting by assignment: This is done by explicitly defining the required type in front of the expression in parenthesis. This can be also considered as forceful casting.
- Syntax:
- (type) expression
- where type indicates the data type to which the final result is converted

- ic not

int("please select exact)

OPERATOR CLASSES -

Explicit conversion

```
// C++ program to demonstrate
                      rror_mod.use_x = False
// explicit type casting
                      rror_mod.use_y = False
                      rror mod.use z = True
#include <iostream>
using namespace std;
                       ob.select= 1
                       er ob.select=1
                       ntext.scene.objects.acti
int main()
                        'Selected" + str(modifies
                        rror ob.select = 0
                        bpy.context.selected_obj
  double x = 1.2;
                        rta.objects[one.name].se
  // Explicit conversion from double to int
                         OPERATOR CLASSES
  int sum = (int)x + 1;
  cout << "Sum = " << sum;
                        vpes.Operator):
                        X mirror to the selected
                       rject.mirror_mirror_x"
  return 0;
                       TOP Y
```

te not

. mod . mirror_obj

- Conversion using Cast operator: A Cast operator is an unary operator which forces one data type to be converted into another data type.
- C++ supports four types of casting:
- Static Cast : int b = static cast<int>(f);

"Selected" + str(modified

X mirror to the selected

is not

rject.mirror_mirror_x"

irror ob.select = 0

- Dynamic Cast
- bpy.context.selected_ob int* c1 = const_cast <int *> (b1); Const Cast
- nt("please select exactle Reinterpret Cast: data_type *var_name = reinterpret_cast <data_type *>(pointer_variable);

polect to mirry

 const_cast is considered safer than simple type casting. It'safer in the sense that the casting won't happen if the type of cast is not same as original object. For example, the following program fails in compilation because 'int *' is being typecasted to 'char *'

```
ob.select= 1
                    er ob.select=1
#include <iostream>
                     text.scene.objects.acti
using namespace std; elected + str(modified)
                    rror ob.select = 0
                    bpy.context.selected_obj
                    ta.objects[one.name].se
int main(void)
                   int("please select exaction
  int a1 = 40:
                      OPERATOR CLASSES
  const int* b1 = &a1;
  char* c1 = const_cast <char *> (b1); // compiler error
  *c1 = 'A';
                       es. Operator):
mirror outputhe selecter
  return 0;
                     prog.cpp: In function 'int main()': prog.cpp:8: error: invalid
                             const cast from type 'const int*' to type 'char*'
```

s se not