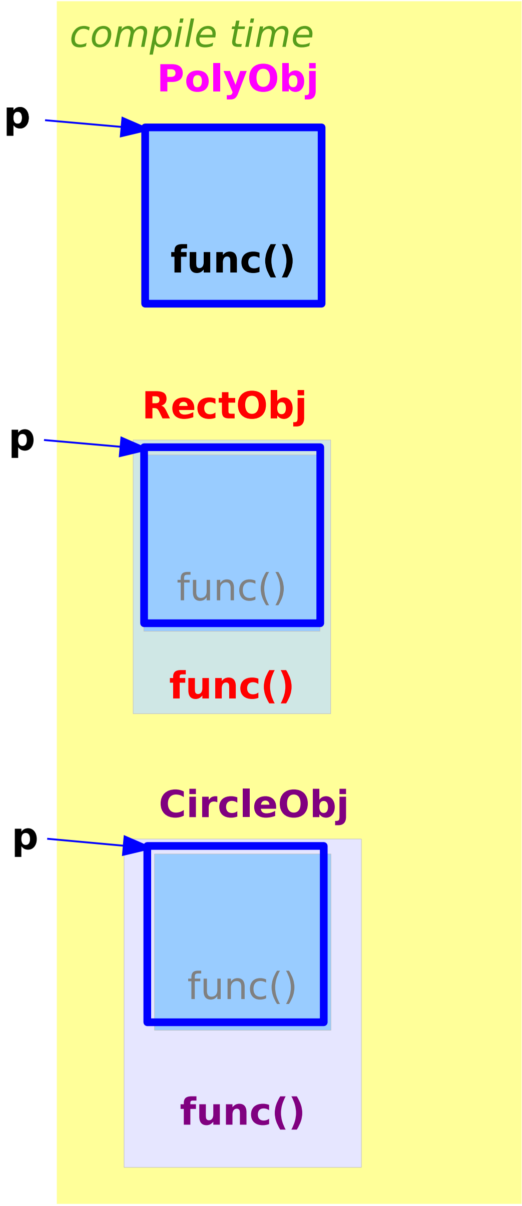
Type Cast (1A)

Young Won Lim
8/16/13

Please send corrections (or suggestions) to youngwlim@hotmail.com.

This document was produced by using OpenOffice.

Dynamic Cast



```
void foo(Poly * p) {

    p can be determined at run time

 Rect   *RectPointer;
 Circle  *CirclePointer;

 RectPointer = dynamic_cast <Rect> (p);

 if (RectPointer != NULL) {

         do specific things pertain to Rect
 }

 CirclePointer = dynamic_cast <Circle> (p);

 if (CirclePointer != NULL) {

         do specific things pertain to Circle
 }


}
```

static_cast & reinterpret_cast (1)

Static Cast

Converts between pointers to related classes

derived classes ⟺ the base class

No safety check during run time

can remove the overhead of run time type checking

non-pointer conversion: standard conversion between fundamental types

```
class A {
public:
  int x;
  int y;
};
```

```
class B {
public:
  float x;
};
```

Reinterpret Cast

Converts any pointer type to any other pointer type, even of unrelated classes.

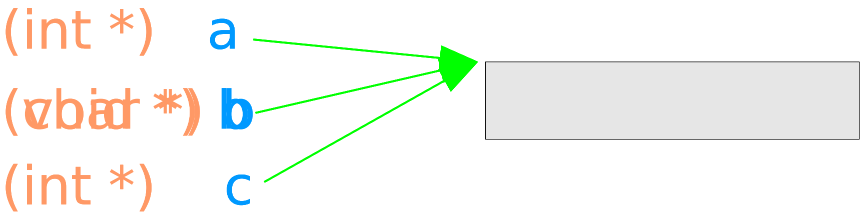
All pointer conversions

No check

```
int main(void) {
  A * a = new A;
  a->x = 10;
  a->y = 20;

  B * b = reinterpret_cast<B*>(a);
  // B * b = static_cast<B*>(a);  error

  cout << a << endl;
  cout << b << endl;
  cout << b->x << endl;

  return 0;
}
```

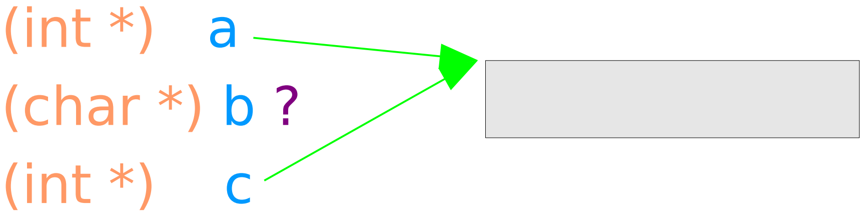
static_cast & reinterpret_cast (2)

```
int     *a  = new int();
char    *b  = static_cast<char*>(a);
int     *c  = static_cast<int*>(b);
```



static_casting a pointer to and from void* preserves the address.

```
int     *a  = new int();
char    *b  = reinterpret_cast<char*>(a);
int     *c  = reinterpret_cast<int*>(b);
```



reinterpret_cast only guarantees that
if you cast a pointer to a different type,
and then reinterpret_cast it back to the original type,
you get the original value.

reinterpret_cast

Reinterpret Cast

pointers ↔ integer types

platform-specific, non-portable

a pointer cast to an integer type large enough to fully contain it, is granted to be able to be cast back to a valid pointer.

platform specific low-level operations

(static cast X)

```
int * p =
     reinterpret_cast<int*>(0x01020304);
```

```
int      *a = new int(1);
char     *b = reinterpret_cast<char*>(a);
int      *c = reinterpret_cast<int*>(b);

printf("%d %x \n", *a, *a);
printf("%d %x \n", *b, *b);
printf("%d %x \n", *c, *c);

cout << a << endl;
cout << b << endl;
cout << c << endl;

int n = reinterpret_cast<int> (a);
int* pointer = reinterpret_cast<int*>( n );


cout << *pointer << endl;
```

const_cast

```
int main (void) {

  const char * const p = "abcde";
  char *q;

  q = const_cast<char *> (p);

  cout << p << endl;

  // *(q+2) = 'X';
  cout << q << endl;

  char s[10] = "XYZ";

  p = const_cast<char *> (s);
  // p = s;
  cout << p << endl;

 return 0;
}
```

References

[1] W Savitch, “Absolute C++”
[2] P.S. Wang, “Standard C++ with objected-oriented programming”
[3] http://www.cplusplus.com
[4] http://stackoverflow.com documents