Queens College, CUNY, Department of Computer Science

Object Oriented Programming in C++

CSCI 211 / 611

Summer 2018

Instructor: Dr. Sateesh Mane

September 15, 2018

Keyword `nullptr`

- This lecture describes the keyword **nullptr.**
- The keyword **nullptr** was introduced in C++11.

1 Pointers & NULL

- We have seen that a pointer can be assigned the value `NULL`.
- The value `NULL` is actually simply the integer 0.
- A pointer can equivalently be assigned the value 0.

```
int *ip;
ip = 0;             // equivalent
ip = NULL;          // equivalent
double *dp;
dp = 0;             // equivalent
dp = NULL;          // equivalent
```

- Nevertheless, a pointer is **not an integer**.
- Hence in C++11, a new keyword **nullptr** has been introduced.
  1. The keyword `nullptr` helps to resolve ambiguities in program compilation.
  2. The keyword `nullptr` does not optimize memory or execution speed.

2 Keyword `nullptr`

- The keyword **nullptr** can be used in place of `NULL`.
  1. Unlike `NULL`, `nullptr` cannot be converted to an integer.
  2. However `nullptr` can be converted to a Boolean value.
- The following program works correctly and prints the expected result.

```
#include <iostream>
using namespace std;

void print(const double *dptr)
{
  cout << "print double *dptr" << endl;
  if (dptr) // returns true if dptr is not nullptr, equivalent to "if (dptr != nullptr)"
    cout << "*dptr = " << *dptr << endl;
  else
    cout << "dptr is nullptr" << endl;
}

int main()
{
  double x = 2.3;
  double *dp = nullptr;                // assigns null value to dp
  print(&x);
  print(dp);
  print(nullptr);
  return 0;
}
```

- In the same way that we can set pointers of different data types to NULL, we can also set pointers of different data types to `nullptr`.

```
#include <iostream>
using namespace std;
// ...

int main()
{
  int *ip = nullptr;
  double *dp = nullptr;

  // etc
  return 0;
}
```

3 Data type nullptr_t

- If `nullptr` is not an integer, what is it?
- In C++11, `nullptr` has its own data type, called **std::nullptr_t.**
  1. We can declare variables of type `nullptr_t`, as in the following program.
  2. We can also declare the input of a function to be of type `nullptr`, as shown below.
  3. The input to such a function can only be a `nullptr`.

```
#include <iostream>
using namespace std;

void print(const double *dptr)
{
  cout << "print double *dptr" << endl;
}

void print(const int *iptr)
{
  cout << "print int *iptr" << endl;
}

void null_func(std::nullptr_t np)      // input can only be nullptr
{
  cout << "print null_func" << endl;
}

int main()
{
  std::nullptr_t np1, np2;  // np1 and np2 are both nullptr
  int *ip = np1;            // equivalent to ip = nullptr
  double *dp = np2;         // equivalent to dp = nullptr
  print(ip);
  print(dp);
  null_func(np1);
  null_func(np2);
  null_func(nullptr);
  return 0;
}
```

- In most cases, we can just use `nullptr` directly.