User Id : weizhe.goh@digipen.edu | Started : 2020.05.27 10:36:19 | Score : 0

DigiPen

References

Practice

References vs pointers

- Both pointers and references let you refer to other memory elements (like variables or structures/objects) indirectly.
- Because there is no such thing as a null reference, C++ requires that references be initialized:

```
int & r; // error!
int n = 0;
int & rn = n; // ok - rn refers to n
```

- Pointers are subject to no such restriction:

```
int * p;
```

So

- If you have a variable whose purpose is to refer to another memory element, but it is possible that there might not be an element to refer to, then that variable must be declared as a pointer
- On the other hand, if the variable must always refer to a memory element (no possibility that the variable is null), you should make the variable a reference.

Efficiency

- The fact that there is no such thing as a null reference implies that it can be more efficient to use references rather than to use pointers. That's because there's no need to test the validity of a reference before using it:

```
void print_int(const int & ri) {
  std::cout << ri << std::endl;
}
```

- Pointers, on the other hand, should generally be tested against null:

```
void print_int(const int * pi) {
  if (pi== nullptr) {
    std::cout << *pi << std::endl;
  }
}
```

Hands-on 1

What is the output of the following code? Guess first and then Run to compare your answer with the correct one.

Run

```
#include<iostream>
int main()
{
  int x = 10;
  int & ref = x;
  ref = 20;
  std::cout << x;
  x = 30;
  std::cout << ref;
  return 0;
}
```

```
2030
```

3

What is the output of the following code? Guess first and then Run to compare your answer with the correct one.

Run

```
#include<iostream>
int main()
{
  int x = 10;
  int y = 20;
  int & r1 = x;
  std::cout << r1;
  int & r2 = y;
  std::cout << r2;
  return 0;
}
```

```
1020
```

Reassignment

- Another important difference between pointers and references is that pointers may be reassigned to refer to different objects.
- A reference, however, always refers to the object with which it is initialized:

```
string s1 = "Tom";
string s2 = "Jane";
string & rs = s1; // rs refers to s1
string * ps = &s1; // ps refers to s1
rs = s2; // rs still refers to s1,
         //   but s1's new value is "Jane"
ps = &s2; // ps now points to s2;
          //  s1 is unchanged
```

References As Parameters

Reference parameters are useful in two cases:

- **Return values.** When a function computes only one value it is considered a better style to return the value with the return statement. However, if a function produces more than one value, it is common to use reference parameters to return values.
- **To pass large structures more efficiently.** This is especially common for passing structs or class objects.
    - If no changes are made to the parameter, it is should be declared const.

```
void swap(File & x, File & y) {
  File temp = x;
  x = y;
  y = temp;
}
```

By signing this document you fully agree that all information provided therein is complete and true in all respects.	Responder sign: