Module 07

Sourangshu Bhattacharya

Objectives & Outlines

Reference variable

Call-by-reference

Swap in C

Swap in C++

const Reference Parameter

Return-by-reference

I/O of a Function

References vs. Pointers

Summary

Module 07: Programming in C++

Reference & Pointer

Sourangshu Bhattacharya

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

sourangshu@cse.iitkgp.ac.in

Slides taken from NPTEL course on Programming in C++

by **Prof. Partha Pratim Das**

Objectives & Outlines

Reference variable

Call-by-reference

Swap in C

Swap in C++

const Reference Parameter

Return-by-reference

I/O of a Function

References vs. Pointers

Summary

Module Objectives

- Understand References in C++
- Compare and contrast References and Pointers

Module Outline

- Reference variable or Alias
  - Basic Notion
  - Call-by-reference in C++
- Example: Swapping two number in C
  - Using Call-by-value
  - Using Call-by-address
- Call-by-reference in C++ in contrast to Call-by-value in C
- Use of `const` in Alias / Reference
- Return-by-reference in C++ in contrast to Return-by-value in C
- Differences between References and Pointers

Module 07

Sourangshu Bhattacharya

Objectives & Outlines

Reference variable

Call-by-reference

Swap in C

Swap in C++

const Reference Parameter

Return-by-reference

I/O of a Function

References vs. Pointers

Summary

Reference

- A reference is an alias / synonym for an existing variable

```
int i = 15;   // i is a variable
int &j = i;   // j is a reference to i
```

`i`	$\leftarrow$ variable
$\boxed{15}$	$\leftarrow$ memory content
200	$\leftarrow$ address (`&i`)
`j`	$\leftarrow$ alias or reference

Program 07.01: Behavior of Reference

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

int main() {
    int a = 10, &b = a; // b is reference of a

    // a and b have the same memory
    cout << "a = " << a << ", b = " << b << endl;
    cout << "&a = " << &a << ", &b = " << &b << endl;

    ++a; // Changing a appears as change in b
    cout << "a = " << a << ", b = " << b << endl;

    ++b; // Changing b also changes a
    cout << "a = " << a << ", b = " << b << endl;

    return 0;
}
```

```
a = 10, b = 10
&a = 002BF944, &b = 002BF944
a = 11, b = 11
a = 12, b = 12
```

- a and b have the same memory location and hence the same value
- Changing one changes the other and vice-versa

Module 07

Sourangshu Bhattacharya

Objectives & Outlines

Reference variable

Call-by-reference

Swap in C

Swap in C++

const Reference Parameter

Return-by-reference

I/O of a Function

References vs. Pointers

Summary

Pitfalls in Reference

Wrong declaration	**Reason**	**Correct declaration**
`int& i;`	no variable to refer to – must be initialized	`int& i = j;`
`int& j = 5;`	no address to refer to as 5 is a constant	`const int& j = 5;`
`int& i = j + k;`	only temporary address (result of `j + k`) to refer to	`const int& i = j + k;`

Module 07

Sourangshu Bhattacharya

Objectives & Outlines

Reference variable

Call-by-reference

Swap in C

Swap in C++

const Reference Parameter

Return-by-reference

I/O of a Function

References vs. Pointers

Summary

C++ Program 07.02: Call-by-reference

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

void Function_under_param_test(// Function prototype
    int &b, // Reference parameter
    int c); // Value parameter

int main() {
    int a = 20;
    cout << "a = " << a << ", &a = " << &a << endl << endl;
    Function_under_param_test(a, a); // Function call

    return 0;
}

void Function_under_param_test(int &b, int c) {  // Function definition
    cout << "b = " << b << ", &b = " << &b << endl << endl;
    cout << "c = " << c << ", &c = " << &c << endl << endl;
}
------- Output -------
a = 20, &a = 0023FA30
b = 20, &b = 0023FA30
c = 20, &c = 0023F95C
```

- Param b is call-by-reference while param c is call-by-value
- Actual param a and formal param b get the same value in called function
- Actual param a and formal param c get the same value in called function
- Actual param a and formal param b get the same value in called function
- However, actual param a and formal param c have *different* addresses in called function

C Program 07.03: Swap in C

Module 07

Sourangshu Bhattacharya

Objectives & Outlines

Reference variable

Call-by-reference

Swap in C

Swap in C++

const Reference Parameter

Return-by-reference

I/O of a Function

References vs. Pointers

Summary

Call-by-value

```
#include <stdio.h>

void swap(int, int); // Call-by-value
int main() {
    int a = 10, b = 15;
    printf("a= %d & b= %d to swap\n", a, b);
    swap(a, b);
    printf("a= %d & b= %d on swap\n", a, b);
    return 0;
}

void swap(int c, int d){
    int t;
    t = c;
    c = d;
    d = t;
}
```

- a= 10 & b= 15 to swap
- a= 10 & b= 15 on swap

- Passing values of a=10 & b=15
- In callee; c = 10 & d = 15
- Swapping the values of c & d
- No change for the values of a & b in caller
- Swapping the value of c & d instead of a & b

Call-by-address

```
#include <stdio.h>

void swap(int *, int *); // Call-by-address
int main() {
    int a=10, b=15;
    printf("a= %d & b= %d to swap\n", a, b);
    swap(&a, &b);
    printf("a= %d & b= %d on swap\n", a, b);
    return 0;
}

void swap(int *x, int *y){
    int t;
    t = *x;
    *x = *y;
    *y = t;
}
```

- a= 10 & b= 15 to swap
- a= 15 & b= 10 on swap

- Passing Address of a & b
- In callee x = Addr(a) & y = Addr(b)
- Values at the addresses is swapped
- Changes for the values of a & b in caller
- It is correct, but C++ has a better way out

Program 07.04: Swap in C & C++

C Program: Call-by-value – wrong

```
#include <stdio.h>

void swap(int, int); // Call-by-value
int main() {
    int a = 10, b = 15;
    printf("a= %d & b= %d to swap\n",a,b);
    swap(a, b);
    printf("a= %d & b= %d on swap\n",a,b);
    return 0;
}

void swap(int c, int d) {
    int t ;
    t = c ;
    c = d ;
    d = t ;
}
```

C++ Program: Call-by-reference – right

```
#include <iostream>
using namespace std;
void swap(int&, int&); // Call-by-reference
int main() {
    int a = 10, b = 15;
    cout<<"a= "<<a<<" & b= "<<b<<"to swap"<<endl;
    swap(a, b);
    cout<<"a= "<<a<<" & b= "<<b<<"on swap"<<endl;
    return 0;
}

void swap(int &x, int &y) {
    int t ;
    t = x ;
    x = y ;
    y = t ;
}
```

C Program: Call-by-value – wrong	C++ Program: Call-by-reference – right
• a= 10 & b= 15 to swap • a= 10 & b= 15 on swap	• a= 10 & b= 15 to swap • a= 15 & b= 10 on swap
• Passing values of a=10 & b=15 • In callee; c=10 & d=15 • Swapping the values of c & d • No change for the values of a & b • Here c & d do not share address with a & b	• Passing values of a = 10 & b = 15 • In callee x = 10 & y = 15 • Swapping the value x & y • Changes the values of a & b • x & y having same address as a & b respectively

Objectives & Outlines

Reference variable

Call-by-reference

Swap in C

Swap in C++

const Reference Parameter

Return-by-reference

I/O of a Function

References vs. Pointers

Summary

Program 07.05: Reference Parameter as `const`

- A reference parameter may get changed in the called function
- Use `const` to stop reference parameter being changed

`const` **reference – bad**

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

int Ref_const(const int &x) {
    ++x;        // Not allowed
    return (x);
}

int main() {
    int a = 10, b;
    b = Ref_const(a);
    cout << "a = " << a <<" and"
         << " b = " << b;
    return 0;
}
```

`const` **reference – good**

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

int Ref_const(const int &x) {

    return (x + 1);
}

int main() {
    int a = 10, b;
    b = Ref_const(a);
    cout << "a = " << a << " and"
         << " b = " << b;
    return 0;
}
```

`const` **reference – bad**	`const` **reference – good**
• Error:Increment of read only Reference 'x'	a = 10 and b = 11
• Compilation Error: Value of x can't be changed • Implies, 'a' can't be changed through 'x'	• No violation.

Objectives & Outlines

Reference variable

Call-by-reference

Swap in C

Swap in C++

const Reference Parameter

Return-by-reference

I/O of a Function

References vs. Pointers

Summary

Program 07.06: Return-by-reference

- A function can return a value by reference
- C uses Return-by-value

Return-by-value

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

int Function_Return_By_Val(int &x) {
    cout <<"x = "<<x<<" &x = "<<&x<<endl;
    return (x);
}
int main() {
    int a = 10;
    cout <<"a = "<<a<<" &a = "<<&a<<endl;

    const int& b = // const needed. Why?
        Function_Return_By_Val(a);

    cout <<"b = "<<b<<" &b = "<<&b<<endl;
    return 0;
}
```

```
a = 10 &a = 00DCFD18
x = 10 &x = 00DCFD18
b = 10 &b = 00DCFD00
```

- Returned variable is temporary
- Has a different address

Return-by-reference

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

int& Function_Return_By_Ref(int &x) {
    cout <<"x = "<<x<<" &x = "<<&x<<endl;
    return (x);
}
int main() {
    int a = 10;
    cout <<"a = "<<a<<" &a = "<<&a<<endl;

    const int& b = // const optional
        Function_Return_By_Ref(a);

    cout <<"b = "<<b<<" &b = "<<&b<<endl;
    return 0;
}
```

```
a = 10 &a = 00A7F8FC
x = 10 &x = 00A7F8FC
b = 10 &b = 00A7F8FC
```

- Returned variable is an alias of a
- Has the same address

Program 07.07: Return-by-reference can get tricky

Return-by-reference

```
#include <iostream>
using namespace std;
int& Return_ref(int &x) {


    return (x);
}

int main() {
    int a = 10, b;
    b = Return_ref(a);
    cout << "a = " << a << " and b = "
         << b << endl;

    Return_ref(a) = 3; // Changes
                       // reference
    cout << "a = " << a;

    return 0;
}
```

```
a = 10 and b = 10
a = 3
```

- Note how a value is assigned to function call
- This can change a local variable

Return-by-reference – Risky!

```
#include <iostream>
using namespace std;
int& Return_ref(int &x) {
    int t = x;
    t++;
    return (t);
}

int main() {
    int a = 10, b;
    b = Return_ref(a);
    cout << "a = " << a << " and b = "
         << b << endl;

    Return_ref(a) = 3;

    cout << "a = " << a;

    return 0;
}
```

```
a = 10 and b = 11
a = 10
```

- We expect a to be 3, but it has not changed
- It returns reference to local. This is risky

Objectives & Outlines
Reference variable
Call-by-reference
Swap in C
Swap in C++
const Reference Parameter
Return-by-reference
I/O of a Function
References vs. Pointers
Summary

I/O of a Function

- In C++ we can changes values with a function as follows:

Orifice	**Purpose**	**Mechanism**
Value Parameter	Input	Call-by-value
Reference Parameter	In-Out	Call-by-reference
`const` Reference Parameter	Input	Call-by-reference
Return Value	Output	Return-by-value Return-by-reference

Recommended Mechanisms

- Call
  - Pass parameters of built-in types by value
    - Recall: Array parameters are passed by reference in C
  - Pass parameters of user-defined types by reference
    - Make a reference parameter `const` if it is not used for output
- Return
  - Return built-in types by value
  - Return user-defined types by reference
    - Return value is not copied back
    - May be faster than returning a value
    - Beware: Calling function can change returned object
    - Never return a local variables by reference

Objectives & Outlines
Reference variable
Call-by-reference
Swap in C
Swap in C++
const Reference Parameter
Return-by-reference
I/O of a Function
References vs. Pointers
Summary

Difference between Reference and Pointer

Pointers	**References**
• Refers to an address	• Refers to an address
• Pointers can point to NULL. int *p = NULL; // p is not pointing	• References cannot be NULL int &j ; //wrong

• Pointers can point to different variables at different times
 	
`int a, b, *p;`
`p = &a; // p points to a`
`...`
`p = &b // p points to b`
	• For a reference, its referent is fixed
	
	`int a, c, &b = a; // Okay`
	`........`
	`&b = c             // Error`

• NULL checking is required	• Makes code faster Does not require NULL checking
• Allows users to operate on the address – diff pointers, increment, etc.	• Does not allow users to operate on the address. All operations are interpreted for the referent
• Array of pointers can be defined	• Array of references not allowed

Module 07

Sourangshu Bhattacharya

Objectives & Outlines

Reference variable

Call-by-reference

Swap in C

Swap in C++

const Reference Parameter

Return-by-reference

I/O of a Function

References vs. Pointers

Summary

Module Summary

- Introduced reference in C++
- Studied the difference between call-by-value and call-by-reference
- Studied the difference between return-by-value and return-by-reference
- Discussed the difference between References and Pointers