



Module 07

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 07: Programming in C++

Reference & Pointer

Partha Pratim Das

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

ppd@cse.iitkgp.ernet.in

Tanwi Mallick
Srijoni Majumdar
Himadri B G S Bhuyan



Module Objectives

Module 07

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

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



Module Outline

Module 07

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

- 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: Lecture 10

Module 07

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

- 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



Reference

Module 07

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

- 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	← variable
15	← memory content
200	← address
j	← alias or reference



Program 07.01: Behavior of Reference

Module 07

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

```
#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



Pitfalls in Reference

Module 07

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

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



C++ Program 07.02: Call-by-reference

Module 07

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

```
#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

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

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++

Module 07

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

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 ;
}
```

- 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
- Here c & d do not share address with a & b

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 ;
}
```

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

- 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



Module 07: End Of Lecture 10

Module 07

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

- 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



Module 07: Lecture 11

Module 07

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

- Use of `const` in Alias / Reference
- Return-by-reference in C++ in contrast to Return-by-value in C
- Differences between References and Pointers



Program 07.05: Reference Parameter as const

Module 07

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

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

const reference – bad	const reference – good
<pre>#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; }</pre>	<pre>#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; }</pre>
<ul style="list-style-type: none"> • Error: Increment of read only Reference 'x' 	<p>a = 10 and b = 11</p>
<ul style="list-style-type: none"> • Compilation Error: Value of x can't be changed • Implies, 'a' can't be changed through 'x' 	<ul style="list-style-type: none"> • No violation.



Program 07.06: Return-by-reference

Module 07

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

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

Return-by-value	Return-by-reference
<pre>#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; }</pre>	<pre>#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; }</pre>
<pre>a = 10 &a = 00DCFD18 x = 10 &x = 00DCFD18 b = 10 &b = 00DCFD00</pre>	<pre>a = 10 &a = 00A7F8FC x = 10 &x = 00A7F8FC b = 10 &b = 00A7F8FC</pre>
<ul style="list-style-type: none"> • Returned variable is temporary • Has a different address 	<ul style="list-style-type: none"> • Returned variable is an alias of a • Has the same address



Program 07.07: Return-by-reference can get tricky

Module 07

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

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



I/O of a Function

- In C++ we can change 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

Module 07

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



Recommended Mechanisms

Module 07

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

- 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



Difference between Reference and Pointer

Module 07

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

Pointers	References
<ul style="list-style-type: none">• Refers to an address	<ul style="list-style-type: none">• Refers to an address
<ul style="list-style-type: none">• Pointers can point to NULL. <code>int *p = NULL; // p is not pointing</code>	<ul style="list-style-type: none">• References cannot be NULL <code>int &j ; //wrong</code>
<ul style="list-style-type: none">• Pointers can point to different variables at different times <code>int a, b, *p;</code> <code>p = &a; // p points to a</code> <code>...</code> <code>p = &b // p points to b</code>	<ul style="list-style-type: none">• For a reference, its referent is fixed <code>int a, c, &b = a; // Okay</code> <code>.....</code> <code>&b = c // Error</code>
<ul style="list-style-type: none">• NULL checking is required	<ul style="list-style-type: none">• Makes code faster Does not require NULL checking
<ul style="list-style-type: none">• Allows users to operate on the address – diff pointers, increment, etc.	<ul style="list-style-type: none">• Does not allow users to operate on the address. All operations are interpreted for the referent
<ul style="list-style-type: none">• Array of pointers can be defined	<ul style="list-style-type: none">• Array of references not allowed



Module Summary

Module 07

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

- 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



Instructor and TAs

Module 07

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

Name	Mail	Mobile
Partha Pratim Das, <i>Instructor</i>	ppd@cse.iitkgp.ernet.in	9830030880
Tanwi Mallick, <i>TA</i>	tanwimallick@gmail.com	9674277774
Srijoni Majumdar, <i>TA</i>	majumdarsrijoni@gmail.com	9674474267
Himadri B G S Bhuyan, <i>TA</i>	himadribhuyan@gmail.com	9438911655