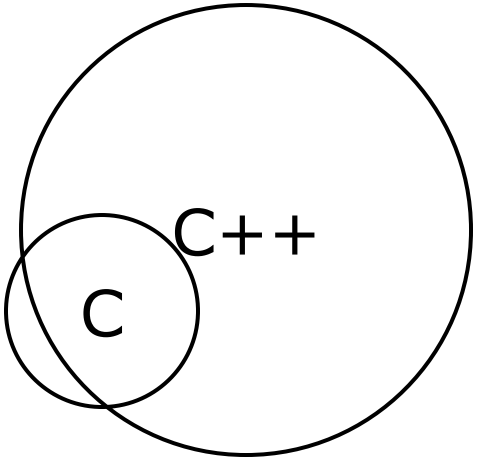
From C to C++

Jiun-Long Huang
National Chiao Tung University

First C++ Program

- Hello, world.

```
#include <stdio.h>
int main()
{
  printf("hello, world");
  return 0;
}
```



- In most situations, C++ is backward compatible to C

First C++ Program (contd.)

```
int main()
{
  float a=3.2;
  int b=5;
  b=a;
  return 0;
}
```

- gcc says nothing
- g++ returns a warning
  - warning: converting to `int' from `float`
- C++ is stricter than C

A More C++-style Program

```cpp
#include <iostream>
using namespace std;
int main()
{
  cout << "hello, world";
  return 0;
}
```

- File size (compiled by gcc/g++)
  - C: 3,072 bytes
  - C++: 276,992 bytes

Differences between C and C++

- Program paradigm
  - C: structural programming
  - C++: object-oriented programming
- C is suitable for the applications that the file size or speed is a major concern
  - Embedded system…
- C++ is more suitable for large and complex applications
- The filenames of C++ programs are usually *.cc or *.cpp

Comments

- C-style comments
  - /* */ ← multi-line
- C++-style comments
  - /* */ ← multi-line
  - // ← single-line
- In C99, C++-style comments are also acceptable

Boolean

- In C, we usually use a non-zero integer to represent true and use 0 to represent false
- C++ supports boolean type: bool
  - C99 also supports boolean type

```
bool isEqual(int a, int b)
{
  if (a==b)
    return true;
  else
    return false;
}
```

Type Casting

- C-style type casting
  - (type)
    - (int) a
- C++ supports C-style type casting
- C++-style type casting
  - static_cast: for standard type casting
    - static_cast<int> (a)

Type Casting (contd.)

- const_cast: for constant type casting
- dynamic_cast: for polymorphic type casting
- reinterpret_cast: for non-standard type casting

Resolving Scope and Variable Declarations

- In C++, :: is used to access global variables

```
#include <iostream>
using namespace std;
int x=1;
int main()
{
  int x=2;
  cout << x;      // 2
  cout << ::x;    // 1
  {
    int x=3;
    cout << x;    // 3
    cout << ::x;  // 1
  }
  return 0;
}
```

Header Files

- C
  - #include <stdio.h>
- Early version of C++
  - #include <stdio.h>
  - #include <iostream.h>
- Current version of C++
  - #include <stdio.h>
  - #include <iostream.h>
  - #include <cstdio>
  - #include <iostream>

C++ Input / Output

- C’s standard I/O functions:
  - scanf(), printf()
- C++ provides new I/O methods
  - stream: a sequence of data (input /output)
  - objects:
    - cin (console input device)
    - cout (console output device)
  - cin and cout are defined within std namespace in the iostream header file

- I/O operations
  - stream insertion operation <<
  - stream extraction operation >>
  - cout << "OOP Using C++";
  - cin >> score;

```
#include <iostream>
using namespace std;
int main()
{
  float length, width, area;
  cout << "Enter length and width ==> ";
  cin >> length >> width;
  area=length*width;
  cout << "Area = " << area;
  return 0;
}
```

- printf and scanf are error-prone

```
int a, b;
scanf("%d %d", &a, &b);
printf("%d %d", a, b);

scanf("%d %s", &a, &b); //Error
printf("%s %d", a, b);  //Error

cin >> a >> b;
cout << a << b;
```

- cin.get()
  - Get a character

```
char ch;
ch=cin.get();
```

- cin.getline (char* s, streamsize n);
  - Get a string

```
char message[50];
cout << "Enter a message: ";
cin.getline(message, 50);
cout << message;
```

C++ Formatting

- Numeric base manipulators
  - dec: sets decimal base
  - hex: sets hexadecimal base
  - oct: sets octal base

```
int x=10, y=100, z=12, q=13, r=100;
cout << hex << x << ' ' << y; // a 64
cout << z << ' ' << dec << q; // c 13
```

C++ Formatting (contd.)

- Character control manipulators
  - endl: insert a new-line character '\n' and flush the buffer
  - ends: insert '\0'
  - flush: flushes the buffer

```
int sp=50;
cout "Speed = " << sp << endl;
```

C++ Formatting (contd.)

- Format control manipulators
  - Should include iomanip
  - setw(int)

```
cout 10 << setw(6) << 20; // 10____20
```

  - setprecision(int)
    - The decimal precision determines the maximum number of digits to be written to express floating-point values.

```
cout<<setprecision(3)<<20.1234; // 20.1
cout<<setprecision(4)<<20.1234; // 20.12
```

C++ Formatting (contd.)

  - setfill(char)

```
cout<<setw(6)<<10;               //    10
cout<<setw(6)<<setfill('$')<<10; //$$$$10
```

- Please study the usage of setiosflags and resetiosflags

Manipulator	Description
Numeric Base Manipulators	
dec	Sets decimal base
hex	Sets hexadecimal base
oct	Sets octal base
Character Control Manipulators	
endl	Inserts a new-line character `\n' and flushes the buffer
ends	Inserts a null character `\0'
flush	Flushes the buffer
Format Control Manipulators	
setw(int)	Sets the field width for a single output field
setprecision(int)	Sets the floating point precision
setiosftag(flag)	Sets the output format flags
resetiosflags(flag)	Resets the output format flags
setfill(char)	Sets the fill character :

Namespaces

- Problem:
  - Errors may occur if duplicate identifiers (names of variables, constants, functions etc.) are used in the same global scope that is shared by all modules.

```
void Init(void)
{
  ...
}
```

a.cc

```
void Init(void)
{
  ...
}
```

b.cc

Namespaces (contd.)

- Example: display functions in allegro game library
  - al_create_display
  - al_destroy_display
  - al_get_new_display_flags
  - al_get_new_display_refresh_rate
  - al_get_new_window_position
  - al_set_new_display_option
  - …

http://alleg.sourceforge.net/

Namespaces (contd.)

- Solution:
  - C++ provides a mechanism called a namespace to prevent such error.
  - The namespace keyword is used to group together logically related programming entities such as variables, objects, functions, and structures.
  - A namespace member identifier is only visible within its namespace.

Namespaces (contd.)

```
namespace namespace_name
{
  //body of the namespace
  //that contains declarations
  //and definitions
}
```

```
namespace a
{
  void Init(void) { ...}
}
```

a.cc

```
namespace b
{
  void Init(void) { ... }
}
```

b.cc

Namespaces (contd.)

```
namespace allegro
{
  create_display {...}
  destroy_display {...}
  get_new_display_flags {...}
  get_new_display_refresh_rate {...}
  get_new_window_position {...}
  set_new_display_option {...}
  ...
}
```

These functions can be further grouped into several classes

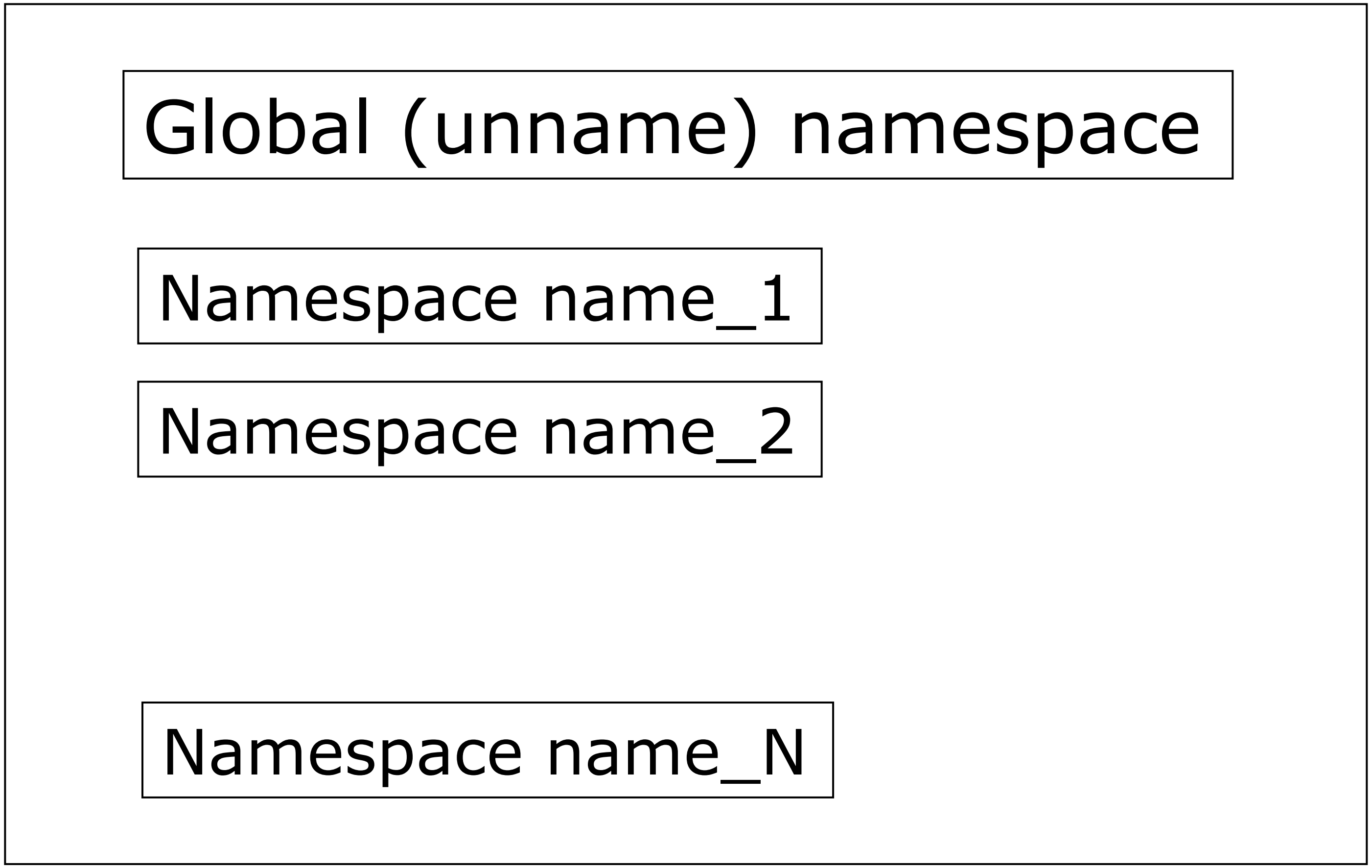
How to Access Namespace Member?

- Accessing namespace members outside the namespace is by preceding a member identifier with its namespace name followed by the scope resolution operator(::).
- A namespace can also be unnamed (anonymous namespace).
  - An unnamed namespace has no identifier.

How to Access Namespace Member? (contd.)

```
namespace Sample
{    //namespace declaration
  int i;
  float f;
  void display()  { cout << i << f ; }
  float getf()  { return f; }
}

Sample::i=33;
Sample::f=1.23;
float x=Sample::getf();
Sample::display();
```



Global (unname) namespace

Namespace name_1

Namespace name_2

Namespace name_N

```
#include <iostream>
using namespace std;
namespace Circle { //named namespace declaration
  const double PI=3.14159265;
  float r; //radius of a circle
  float a; //area of a circle
  float area()
    {return PI*r*r;} //Computes and returns area
  void print( )
    {cout << "Area = " << a << endl;} //Prints area
}
```

```
float a=0; //total area
void print()
  {cout << "\nTotal area = " << a;}

int main()
{
  for(int i=0; i<3; i++)
  {
    cout<<"Enter radius of circle #"<<(i+1)<<": ";

    cin >> Circle::r;
    Circle::a=Circle::area();
    Circle::print(); //Calls print() from Circle
    a=a+Circle::a;
  }
  print();
  return 0;
}
```

using Directive

- Repetitive use of a namespace name followed by :: each time a preceding member of a namespace is listed is often not convenient, particularly if they are used frequently.
- To eliminate this redundant syntax, C++ provides the using directive,
  - using namespace <namespace name>

```
#include <iostream>
namespace Rectangle { //user-defined namespace
  float length;
  float width;
  void area()
    { cout << "Area = " << (length*width); }
}
int main()
{
  std::cout << "Enter length => ";
  std::cin >> Rectangle::length;
  std::cout << "Enter width => ";
  std::cin >> Rectangle:: width ;
  Rectangle::area();
  return 0;
}
```

```
#include <iostream>
using namespace std;  //predefined namespace
namespace Rectangle { //user-defined namespace
  float length;
  float width;
  void area()
    {cout << "Area = " << (length*width); }
}
//Specifies user-defined namespace
using namespace Rectangle;
int main()
{
  cout << "Enter length => ";
  cin >> length;
  cout << "Enter width => ";
  cin >> width ;
  area();
  return 0;
}
```

Nested Namespaces

- □ A namespace can also be declared within another namespace (nested namespaces).
- □ When used within an outer namespace, a member identifier of an inner namespace must be preceded with its namespace name followed by the scope resolution operator.

```
#include <iostream>
using namespace std;
namespace a
{
  int A=1;
  namespace b
  {
    int B=2;
  }
}
int main()
{
  a::A++;
  a::b::B++;
  cout<<a::A<<' '<<a::b::B;
}
```