fit@hcmus

PROGRAMMING TECHNIQUES

Week 1 - Review

Nguyễn Hải Minh - 01/2024

Review

1. Pass by reference vs pass by value
2. Arrays of characters
   - Reading strings using 3 argument `cin.get`
3. Structures, arrays of structures
   - Passing structures by ref vs by value
4. Reading/writing external data files

1. PASS BY REFERENCE VS PASS BY VALUE

Pass by reference vs Pass by value

- Why use **pass by reference**?

```
void print(float & data);
```

  - supply a value back to the calling routine
  - more effective use of memory

- Why use **pass by value**?

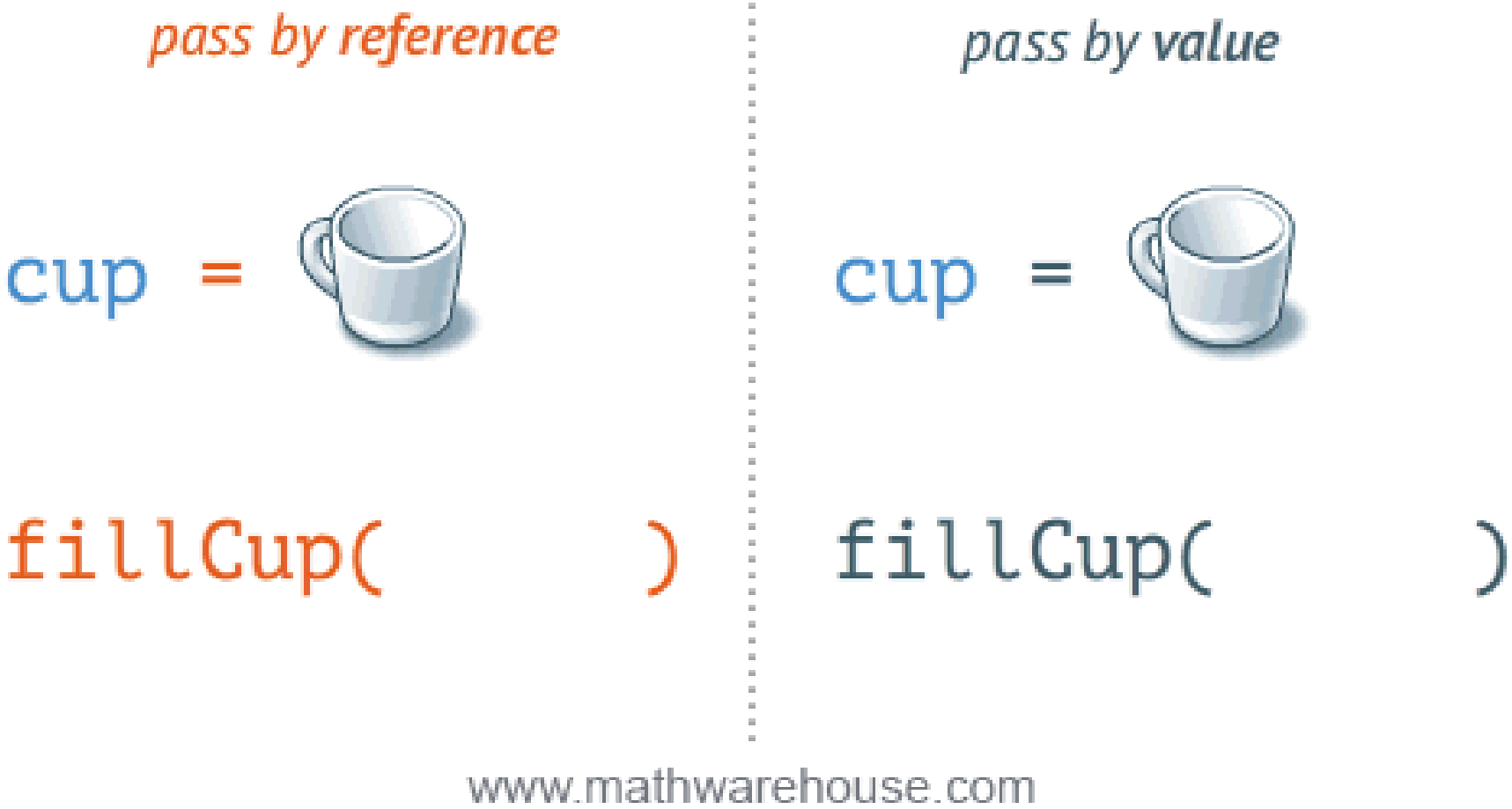
```
void print(float data);
```

  - <u>only</u> when you need a spare and duplicate copy of the data <u>or</u> if passing fundamental data types (like an int, short, char)

- Why use **constant references**?

```
void print(const float & data);
```

Pass by reference vs Pass by value



Pass by Value

```
int sumup(int first, int second);//function prototype

int main() {
    int total, number, count;
    total = 0;
    for (count = 1; count <= 5; count++) {
        cout << " Enter a number to add: ";
        cin >> number;
        total = sumup(total, number);//function call
    }
    cout << " The result is: " << total << endl;
    return 0;
}

int sumup(int first, int second) { //definition
    return  first + second;
}
```

Pass by Reference

```
void convert(float inches, float& mils);

int main() {
    float in;//local variable to hold # inches
    float mm;//local variable for the result
    cout << "Enter the number of inches : ";
    cin >> in;
    convert(in, mm);//function call
    cout << in << "inches converts to " << mm << "mm";
    return 0;
}

void convert(float inches, float& mils) {
    mils = 25.4 * inches;
}
```

Pass by Constant Refences

Constant References

- If you pass an object by reference, the function could change it.
  - Even if the intention was just *to be more efficient.*
- Using the keywork **`const`** ensures that the referenced object can be passed safely.
- The function **cannot** change the object.

```
void convert2(const float& inches, float& mils)
{
    mils = 25.4 * inches;
}
```

Passing Array in Function

- How is an array passed to a function?
  - What does the function call look like?
  - What does the prototype look like?
  - Is there any way to pass an array to a function by value? vs. by reference?
  - It is important to realize that the name of an array is a **constant address of the first element in the array**. It is that which is passed (by value)!!!!
  - How to protect an array with `const`

Passing Array in Function

```
const int ArSize = 8;
int sum_arr(int arr[], int n);

int main(){
   int cookies[ArSize] = {1,2,4,8,16,32,64,128};
   int sum = sum_arr(cookies, ArSize); //function call
   cout << "Total cookies eaten: " << sum << "\n";
   return 0;
}

int sum_arr(int arr[], int n) //return the sum of an integer array
{
   int total = 0;
   for (int i = 0; i < n; i++)
   total = total + arr[i];
   return total;
}
```

```
cookies == &cookies[0]
//array name is address of
the first element
```

Passing Array in Function

```
void show_array(const double ar[], int n);

void show_array(const double ar[], int n)
{
   ar[0] += 10;
}
```

*>> error: cannot modify a const object in function show_array(const double *,int)*

2. ARRAY OF CHARACTERS

Reading string using `cin.get`

Array of characters

- Reading in arrays of characters:
  - What is the advantage/disadvantage of:

```
char s[5];
cin >> s;
```

  - What if the user input:
    - "Hi "

'H'	'i'	'\0'		

    - "Hello"

'H'	'e'	'l'	'l'	'o'	'\0'

✓ Skip white space
✓ Store '\0' after the last character read in

Extremely dangerous!!! segmentation fault or core dump when running

Array of characters

- Reading in arrays of characters:
  - What is the advantage/disadvantage of:

```
char s[5];
cin.get(s,5,'\n');
```

→ Read in the next sequence of characters up until 4 characters are read or the delimiting character is encountered ('\n' by default)

'H'	'e'	'l'	'l'	'\0'	'o\n'

Left in the input buffer!!!

✓ Can read white space
✓ Never store characters outside array bounds

Array of characters

- Reading in arrays of characters:
  - What does this do:

```
while (cin.get() != '\n');
```

or:

```
cin.ignore(100, '\n');
```

✓ Flush the input buffer

Arrays

- What is the purpose of a function <u>declaration (i.e., prototype)</u>
  - To allow a function to be called even if it is defined (i.e., implemented) later or in some other file.
- What about ***<u>defining arrays</u>***,
  - Can the size be variable?

```
int n;
char s[n];
```

NO!!!

  - Remember to allow 1 character in a "string" for the '\0' (terminating nul)

```
char s[5]; //array of 4 characters
```

3. STRUCTURE &

ARRAY OF STRUCTURES

Passing structures by ref vs by value

Structure

```
struct storeitem {
   char item[20];
   float cost;
   float price;
   int barcode;
}; //<-- don't forget the semicolon here

storeitem one_item;
```

Structure in function

```
struct POINT {
   int x;
   int y;
};

//Get x and y from user's input
void Input(POINT& p)
{
   cout << "Please input x : ";
   cin >> p.x;
   cout << "Please input y : ";
   cin >> p.y;
}
```

Pass by Reference

Structure in function

```
//Get x and y from user's input
void Input(POINT p[], int n)
{
   for (int i = 0; i < n; i++)
   {
      cout << "Please input x of p[" << i << "]: ";
      cin >> p[i].x;
      cout << "Please input y of p[" << i << "]: ";
      cin >> p[i].y;
   }
}
```

Pass by Value or Reference?

Structure in function

```
//Print out p.x; p.y to the screen
void Show(POINT p)
{
    cout << "x = " << p.x << "; y = " << p.y << endl;
}
//Return a new POINT which has the same y with p
//but x is zero
POINT NewPoint(POINT p)
{
    POINT np = p;
    np.x = 0;
    return np;
}
```

Pass by Value

Pass by Value

4. READ & WRITE EXTERNAL FILE

Read an External File

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
   ifstream in;
   in.open("test.dat"); //Open file to read
   if (in) {
      char ch = in.get();//can also use in >> ch;
      while (!in.eof()) {
         cout << ch;
         ch = in.get();
      }
      in.close(); //Close file
   }
}
```

Read an External File

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
   ifstream in;
   in.open("test.dat"); //Open file to read
   if (in) {
      char first_line[81], second_line[81];
      in.get(first_line, 81, '\n');
      in.get();//eat the carriage return
      in.get(second_line, 81, '\n');
      in.close(); //Close file
   }
}
```

- ✓ Read more than 1 single line
- ✓ Remember to eat the carriage return

Write to an External File

```
#include <fstream>
#include <iostream>
using namespace std;
int main()
{
   ofstream out;
   out.open("test.dat", ios::app); //Open file to write
   if (out) {
      out << "Hello World!";
      out.put('!');
      out << endl;
      out.close(); //Close file
   }
}
```

Next week's topic

- Pointers & Dynamic Memory
- Homework:
  - Read your textbook: C++ Primer Plus page 153~


Q&A