03. Functions, Arrays, Strings and Parameter Passing - 01

Oritented Object Programming C++: Chapter 03

- Time: 120 Minutes
- Instructor: Thang, Nguyen Chien
- Email: chienthangplus@gmail.com
- Phone: 0349 688 571

Previous lessons

1. Introduction C++
   - C++ vs C
   - Compilers, IDEs
2. C++ Language Basics
   - Types: `int`, `float`, `double`
   - Control Structures: `if`, `for`, `while`, `switch-case`…

Contents

Objectives

- Explain how a function works
- Explain function prototyping
- Demonstrate how function overloading
- Determine how default arguments are used with functions
- Explain recursion with the help of an example
- Explain array initialization with example
- Demonstrate how passing arrays to functions

I. Functions

- A function is a set of instructions that can perform a specific task, and can be called by others.
- Benefits:
  - Reusability
  - Modularity
  - Overall programming simplicity
- Functions are classified into:
  - Library functions
  - User-defined functions

I. Functions (cont.)

- Function **definition**

```
<return type> <function name>(<argument list>)
{
    <body of the function>
}
```

- `<return type>` is `void` when the function return no values
- `return` statement used in the body to exit a function and return a value.
- In `void` functions, do not need the `return`
- `<argument list>`: can be empty

I. Functions (cont.)

- Example 01:

```
void showRectangle() {
   cout << "****************" << endl;
   cout << "*              *" << endl;
   cout << "*              *" << endl;
   cout << "****************" << endl;
}
int main() {
   showRectangle();
   return 0;
}
```

Output

```
****************
*              *
*              *
****************
```

I. Functions (cont.)

❖ Example 02:

```
int maxTwoNumbers(int x, int y) {
    if (x > y)
        return x;
    return y;
}
int main() {
    int bigger, a = 2, b = 10;
    bigger = maxTwoNumbers(a, b);
    cout << "The bigger number is " << bigger;
    return 0;
}
```

Output

```
The bigger number is 10
```

I. Functions (cont.)

- Example 02:

```
int maxTwoNumbers(int x, int y) {
    if (x > y)
        return x;
    return y;
}
int main() {
    int bigger, a = 2, b = 10;
    bigger = maxTwoNumbers(a, b);
    cout << "The bigger number is " << bigger;
    return 0;
}
```

b	10
a	2
bigger	???

I. Functions (cont.)

- Example 02:

```
int maxTwoNumbers(int x, int y) {
    if (x > y)
        return x;
    return y;
}
int main() {
    int bigger, a = 2, b = 10;
    bigger = maxTwoNumbers(a, b);
    cout << "The bigger number is " << bigger;
    return 0;
}
```

b	**10**
a	**2**
bigger	**???**

I. Functions (cont.)

- Example 02:

```
int x = a, int y = b
```

```
int maxTwoNumbers(int x, int y) {
    if (x > y)
        return x;
    return y;
}
int main() {
    int bigger, a = 2, b = 10;
    bigger = maxTwoNumbers(a, b);
    cout << "The bigger number is " << bigger;
    return 0;
}
```

y	**10**
x	**2**
b	**10**
a	**2**
bigger	**???**

I. Functions (cont.)

- Example 02:

```
int maxTwoNumbers(int x, int y) {
    if (x > y)
        return x;
    return y;
}
int main() {
    int bigger, a = 2, b = 10;
    bigger = maxTwoNumbers(a, b);
    cout << "The bigger number is " << bigger;
    return 0;
}
```

y	10
x	2
b	10
a	2
bigger	???

I. Functions (cont.)

- Example 02:

```
int maxTwoNumbers(int x, int y) {
    if (x > y)
        return x;
    return y;
}
int main() {
    int bigger, a = 2, b = 10;
    bigger = maxTwoNumbers(a, b);
    cout << "The bigger number is " << bigger;
    return 0;
}
```

....	10
y	10
x	2
b	10
a	2
bigger	???

I. Functions (cont.)

- Example 02:

```
int maxTwoNumbers(int x, int y) {
    if (x > y)
        return x;
    return y;
}
int main() {
    int bigger, a = 2, b = 10;
    bigger = maxTwoNumbers(a, b);
    cout << "The bigger number is " << bigger;
    return 0;
}
```

....	10
y	10
x	2
b	10
a	2
bigger	???

I. Functions (cont.)

- Example 02:

```
int maxTwoNumbers(int x, int y) {
    if (x > y)
        return x;
    return y;
}
int main() {
    int bigger, a = 2, b = 10;
    bigger = maxTwoNumbers(a, b);
    cout << "The bigger number is " << bigger;
    return 0;
}
```

....	**10**

b	**10**
a	**2**
bigger	**10**

I. Functions (cont.)

- Example 02:

```
int maxTwoNumbers(int x, int y) {
    if (x > y)
        return x;
    return y;
}
int main() {
    int bigger, a = 2, b = 10;
    bigger = maxTwoNumbers(a, b);
    cout << "The bigger number is " << bigger;
    return 0;
}
```

Output

```
The bigger number is 10
```

b	**10**
a	**2**
bigger	**10**

I. Functions (cont.)

- Function **Declaration**
  - Describes the function interface to the compiler
  - When a function is called
    - The compiler uses the template to ensure that proper arguments are passed, and the return value is correctly

```
<return type> <function name>(<argument list>);
```

- Examples:
  - `int maxTwoNumbers(int x, int y);`
  - `int maxTwoNumbers(int, int);`
  - `void printOddNumbersFrom(int k);`

I. Functions (cont.)

- Example 03:

```
int maxTwoNumbers(int x, int y);
int main() {
    int a = 2, b = 10;
    int bigger = maxTwoNumbers(a, b);
    cout << "The bigger number is " << bigger;
    return 0;
}
int maxTwoNumbers(int x, int y) {
    if (x > y)
        return x;
    return y;
}
```

Contents

II. Function Overloading

- Using the same function name but ***different parameters*** to create functions that perform a variety of different tasks
- Example: get average of 2, 3 or 4 numbers

```
float getAverage(float a, float b) {
    return (a + b) / 2;
}
float getAverage(float a, float b, float c) {
    return (a + b + c) / 3;
}
float getAverage(float a,float b, float c,float d) {
    return (a + b + c + d) / 4;
}
```

II. Function Overloading (cont.)

```cpp
int main() {
  // invoke getAverage(float a, float b)
  cout << getAverage(1.2, 4.5) << endl;
  // invoke getAverage(float a, float b, float c)
  cout << getAverage(1.6, 3.6, -7.5) << endl;
  // invoke:
  // getAverage(float a, float b, float c, float d)
  cout << getAverage(4.5, 1.5, 6.5, 1.9)<< endl;
  return 0;
}
```

Output

```
2.85
-0.766667
3.6
```

Contents

III. Default Arguments

- In C++ functions:
  - Arguments can have **default values** from right to left
  - **Default values** are specified when the function is declared.
  - The function assigns a default value to the parameter which does not have a matching argument in the function call
- Example:

```
int sum(int a, int b, int c = 0); // OK
int sum(int a, int b = 0, int c); // Not OK
int sum(int a, int b = 0, int c = 0);// OK
int sum(int a = 0, int b, int c = 0);//Not OK
```

III. Default Arguments (cont.)

- Example 01:

```
float getAverage(float a, float b, float c = 0,
                                   float d = 0) {
    return (a + b + c + d) / 4;
}
int main() {
  cout << getAverage(1.2, 4.5) << endl;
  cout << getAverage(1.6, 3.6, -7.5) << endl;
  cout << getAverage(4.5, 1.5, 6.5, 1.9)<< endl;
  return 0;
}
```

getAverage(1.2, 4.5, 0, 0)

getAverage(1.2, 4.5, -7.5, 0)

Output

```
2.85
-0.766667
3.6
```

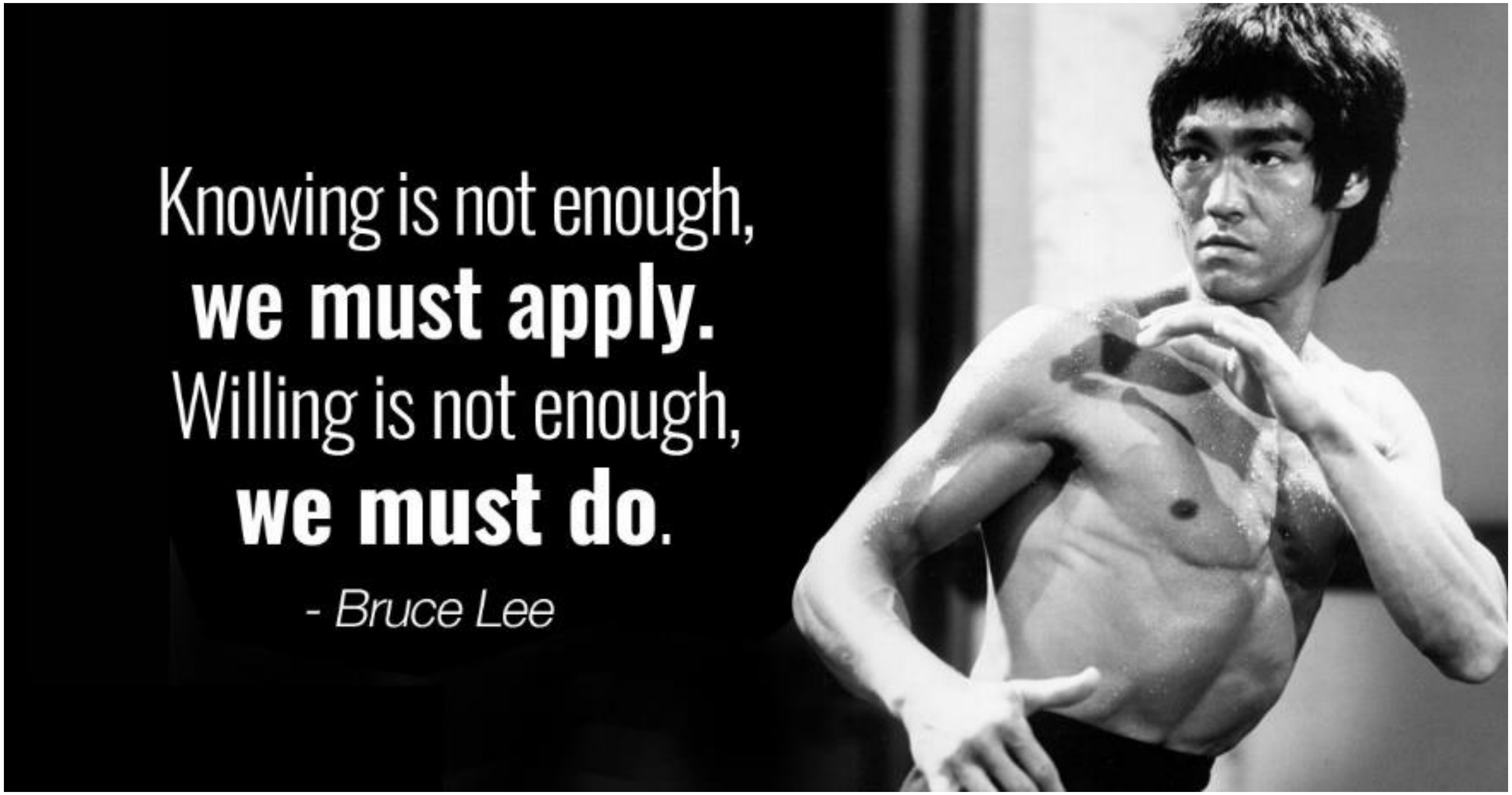
III. Default Arguments (cont.)

- Example 02:

```
float getAverage(float, float, float = 0, float = 0);
int main() {
   cout << getAverage(1.2, 4.5) << endl;
   cout << getAverage(1.6, 3.6, -7.5) << endl;
   cout << getAverage(4.5, 1.5, 6.5, 1.9)<< endl;
   return 0;
}
float getAverage(float a, float b, float c, float d)
{
      return (a + b + c + d) / 4;
}
```

Contents

Practices by Your-Self in 10 minutes



Review 01

```
int Fibonacci(int n) {
    if (n == 1 || n == 2)
        return 1;
    return Fibonacci(n - 1) + Fibonacci(n - 2);
}
int main() {
    cout << Fibonacci(5) << endl;
    return 0;
}
```

Output

```
5
```

Review 02

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

Output

```
2 | 2
2 | 3
2 | 4
```

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

a	**1**

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

sa	1
a	1

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

sa	**1**
a	**2**

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

sa	**2**
a	**2**

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

Output

```
2 | 2
```

sa	**2**
a	**2**

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

Output

```
2 | 2
```

sa	**2**
a	**2**

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

Output

```
2 | 2
```

sa	**2**

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

Output

```
2 | 2
```

sa	**2**
a	**1**

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

Output

```
2 | 2
```

sa	**2**
a	**2**

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

Output

```
2 | 2
```

sa	**3**
a	**2**

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

Output

```
2 | 2
2 | 3
```

sa	3
a	2

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

Output

```
2 | 2
2 | 3
```

sa	**3**
a	**2**

Review 02 (cont.)

```
void func() {
    int a = 1;
    static int sa = 1;
    a = a + 1;
    sa = sa + 1;
    cout << a << " | " << sa << endl;
}
int main() {
    func();
    func();
    func();
    return 0;
}
```

Output

```
2 | 2
2 | 3
2 | 4
```

sa	**4**

Review 03

```
void func(int a, int b, int c) {
    cout << a << " " << b << " " << c << endl;
}
int main() {
    int i = 1;
    func(i++, i++, i++);
    return 0;
}
```

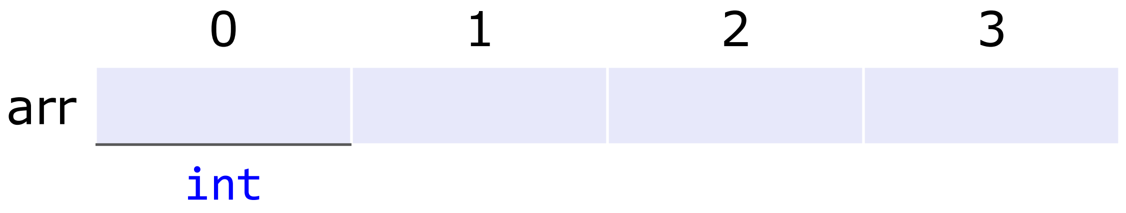
Output

```
3 2 1
```

Contents

IV. Arrays

- An array in C++ is a collection of elements
  - Elements stored at contiguous memory locations
  - Elements can be accessed randomly using indices of an array.
- Declaration for an array: `type` name [`elements`];
- Example:
  - `int arr[4];`



IV. Arrays (cont.)

- Initializing arrays:
  - `int arr1[4] = {5, 7, 1, 2};`

	0	1	2	3
arr1	5	7	1	2

  - `int arr2[4] = {9, 2};`

	0	1	2	3
arr2	9	1	0	0

  - `int arr3[] = {7, 3, 5, 1, 2};`

	0	1	2	3	4
arr3	7	3	5	1	2

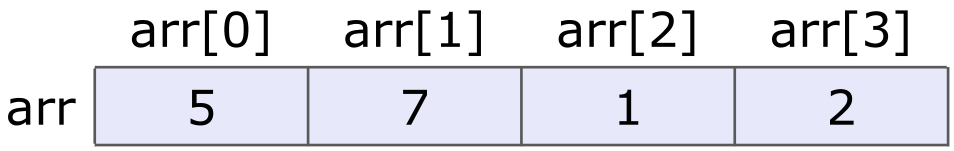
IV. Arrays (cont.)

- Array elements are accessed by using an integer index.
  - Array index starts with 0 and goes till size of array minus 1
  - Syntax: name [`index`];
- Example:

```cpp
int arr[4] = { 5, 7, 1, 2 };
for (int i = 0; i < 4; i++) {
    cout << arr[i] << " ";
}
```

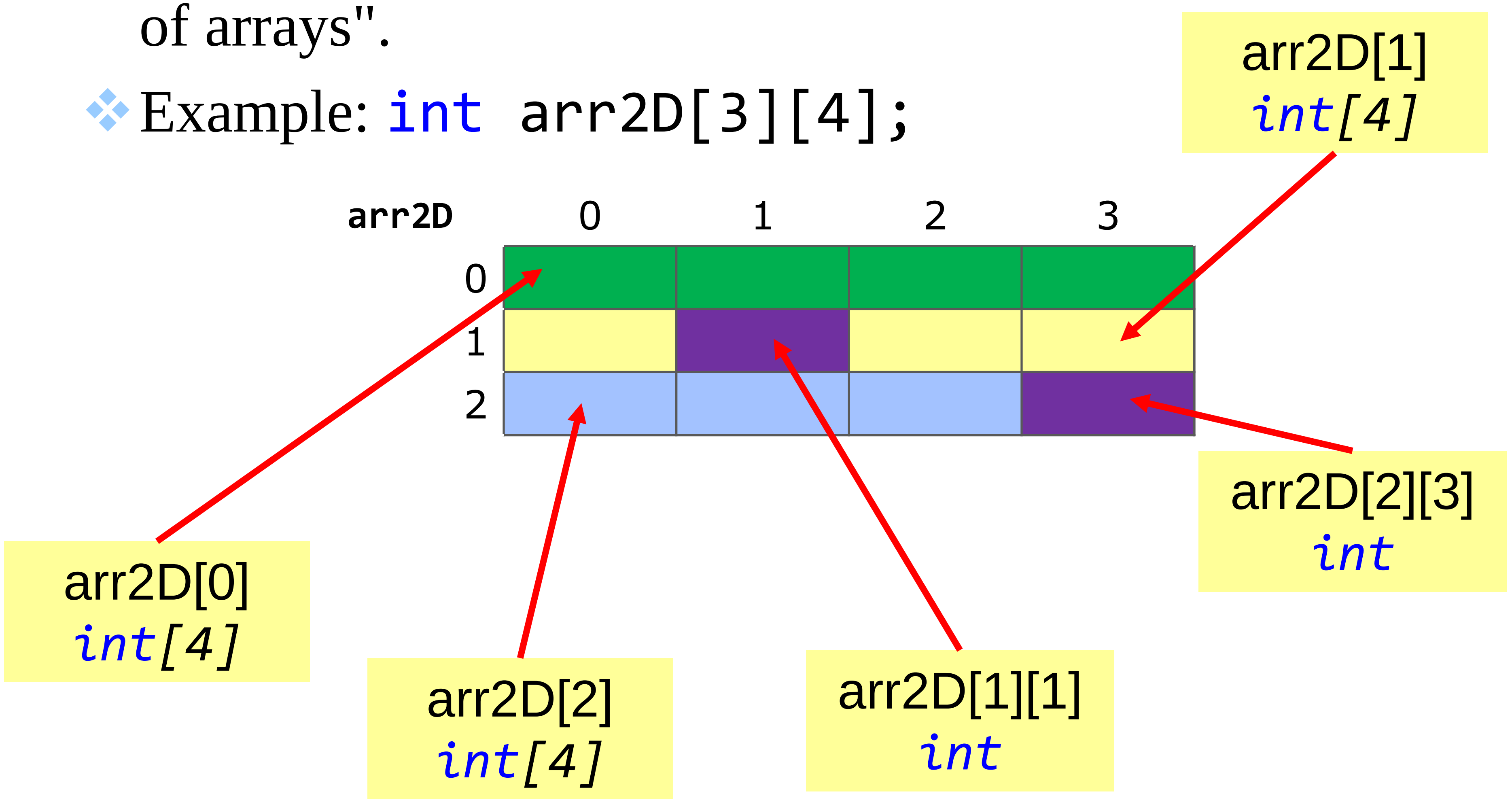
Output

```
5 7 1 2
```



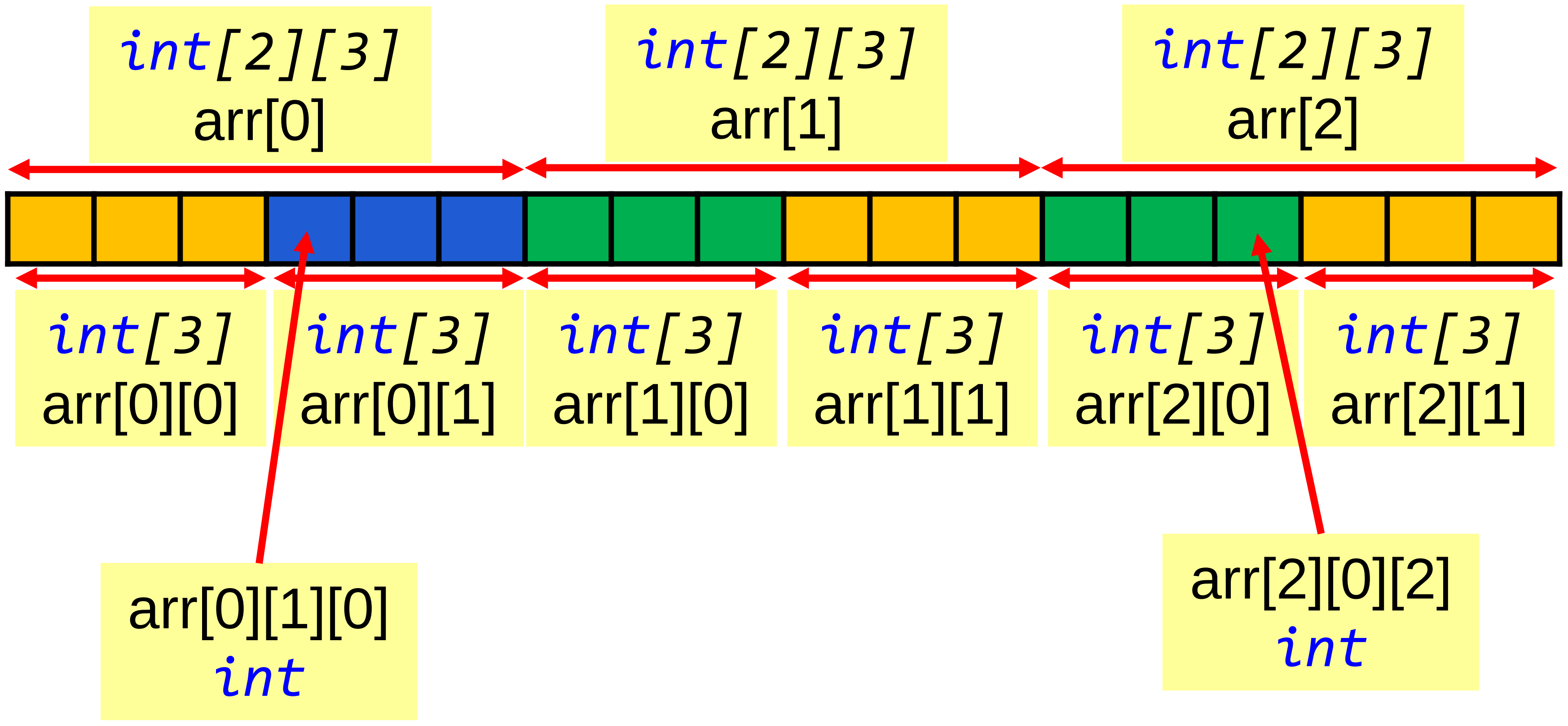
IV. Arrays (cont.)

- Multidimensional arrays can be described as "arrays of arrays".
- Example: `int arr2D[3][4];`



IV. Arrays (cont.)

- Multidimensional arrays can be described as "arrays of arrays".
- Example: `int arr[3][2][3];`



IV. Arrays (cont.)

- Initializing multidimensional arrays

```
int arr2D[2][3] = { {3, 2, 4}, {8, 2, 5} };
for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 3; j++) {
        cout << arr2D[i][j] << " ";
    }
  cout << endl;
}
```

arr2D

	0	1	2
0	3	2	4
1	8	2	5

Output

```
3 2 4
8 2 5
```

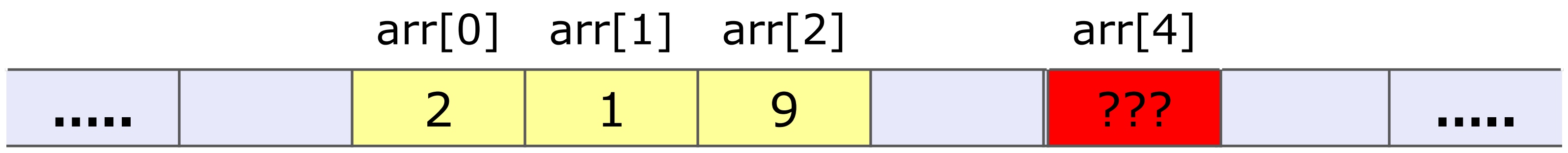
IV. Arrays (cont.)

- Example 01: Accessing an out-of-bounds value

```
int main() {
    int arr[3] = { 2, 1, 9 };
    cout << arr[4];
    return 0;
}
```

Output

```
11533456
```



IV. Arrays (cont.)

- Example 02: Initializing multidimensional arrays

```
int arr2D[3][3] = { {3, 2}, {8} };
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
        cout << arr2D[i][j] << " ";
    }
  cout << endl;
}
```

arr2D	0	1	2
0	3	2	0
1	8	0	0
2	0	0	0

Output

```
3 2 0
8 0 0
0 0 0
```

IV. Arrays (cont.)

❖ Example 03: Passing arrays to a function

```
int sum(int arr[], int size) {
    int s = 0;
    for (int i = 0; i < size; i++)
        s += arr[i];
    return s;
}
int main() {
    int arr[3] = { 2, 4, 6 };
    int size = sizeof(arr) / sizeof(arr[0]);
    cout << "size: " << size << endl;
    cout << "sum: " << sum(arr, size);
    return 0;
}
```

Output

```
size: 3
sum: 12
```

IV. Arrays (cont.)

❖ Example 04: Passing multidimensional arrays

```
int sum(int arr[][3][2], int size) {
    int s = 0;
    ...
    return s;
}
int main() {
    int arr[3][3][2] = {{{1,2},{3,2},{1,5}},
{{7,2},{8,1}} };
    cout << "sum: " << sum(arr, 5);
    return 0;
}
```

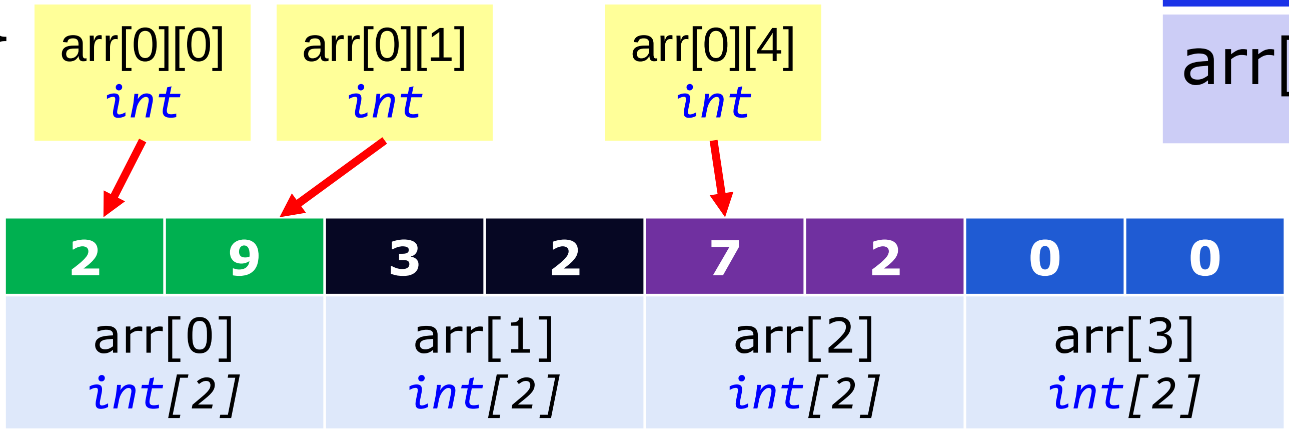
IV. Arrays (cont.)

- Example 05: Accessing an out-of-bounds value

```
int main() {
    int arr[4][2] = {{2,9}, {3,2}, {7,2}};
    cout << "arr[0][4]: " << arr[0][4];
    return 0;
}
```

Output

```
arr[0][4]: 7
```



Contents

Summary

- Function declaration and definition likes C
- **Function overloading**: functions have a same name
- Arguments have default values, the values will be assigned for parameters which does not have a matching argument in function call
- An array in C++ is a collection of elements
  - Elements stored at contiguous memory locations
  - Elements can be accessed randomly using indices of an array.
- Multidimensional arrays can be described as "arrays of arrays".

Problems

1. Write a function to get the largest values in an array
2. Write input(…) and display(…) functions to get values from the keyboard and display them in to the screen.
3. Write a function to display all prime numbers in an array.
4. Write iSort(…) function to sort elements increasing in an array
5. Write iInsert(…) function to insert an element to a sorted array.