Introduction to C++

Georgia Koutsandria

Internet of Things A.Y. 18-19

Prof. Chiara Petrioli
Dept. of Computer Science
Sapienza University of Rome

PART II

Functions

Functions Basics

- A *function* is a block of code with a name.
- It is executed by calling the given name, and it can be called from some point of the program.
- Common syntax:

```
type name(parameter1, parameter2, …){statements}
```

Functions Basics-An example

- Determine the factorial of a given number. The factorial of a number *n* is the product of the numbers from 1 through *n*. The factorial of 5, for example, is 120:

$$1 * 2 * 3 * 4 * 5 = 120$$

```
int fact(int val)
{
        int ret = 1;
        while (val > 1)
                ret *= val--;
        return ret;
}
```

Functions Basics

returns an `int` value

Function named `fact`

takes one `int` parameter

```
int fact(int val)
{
        int ret = 1; //local variable to hold the result
        while (val > 1)
                ret *= val--; //ret=ret*val and decrement val

        return ret; //return the result
}
```

postfix decrement operator (--)

Ends execution of `fact` and returns the value of ret

Calling a Function

- A *function* call
  - Initializes the parameters from the arguments
  - Transfers control to that *function.*
- Execution of the called *function* begins.

```
int main(){
      int j = fact(5);
      cout << "5! is " << j << endl;
      return 0;
}
```

Calling a Function

- To call *fact*, we must supply an `int` value.
- The result of the call is also an `int`.

```
int main(){
      int j = fact(5);
      cout << "5! is " << j << endl;
      return 0;
}
```

Example

```
//function example
#include <iostream>
using namespace std;
int addition(int a, int b){
      int r;
      r=a+b;
      return r;
}
int main(){
      int z;
      z = addition(5,3);
      cout << "The result is " << z << ".\n ";

}
```

Functions with no type

- When a function does not need to return a value, the type to be used is `void`.
- This is a special type to represent the absence of value.
- The `void` can also be used in the function's parameter list to specify that the function takes no actual parameters when called.

```
printmessage();
```

Note the use of the empty pair of parentheses!

Passing arguments by value

- Arguments can be passed by value

```
int x=5, y=3, z;
z = addition (x,y);
```

- Only copies of the variables values at that moment are passed to the function.

- Modifications on the values of the variables have not effect on the values of the variables outside the function.

Passing arguments by reference

- Access an external variable from within a function.
- The variable itself is passed to the function.
- Any modification on the local variables within the function are reflected in the variables passed as arguments in the call.
- References are indicated with an ampersand (&) following the parameter type.

Passing arguments by reference

```
//passing parameters by reference
#include <iostream>
using namespace std;
void duplicate(int& a, int& b){
      a*=2;
      b*=3;
}
int main(){
      int x=1, y=3;
      duplicate(x, y);
      cout << "x=" << x << ", y=" << y << "\n";
      return 0;
}
```

Declaring functions

- Functions cannot be called before they are declared.
- Functions should be declared before calling `main`.
- If the `main` is defined before an undeclared function is called, then the compilation of the program will fail.

```
int fact(int a, int b);
void even(int x);
```

Function declaration

Recursivity

- The property that functions have to be called by themselves.
- It can be useful for specific tasks such as sorting, factorial etc..
- E.g.: Calculate the factorial of number 3 (3!= 1 * 2 * 3 = 6) using a recursive funtion.

Recursivity

```
//factorial
#include <iostream>
using namespace std;
long factorial(long a){
      if (a > 1)
            return (a * factorial (a-1));
      else
            return 1;
}
int main(){
      long number = 3;
      long fact;
      fact = factorial(number);
      cout << number << "! = " << fact << "\n ";

}
```

Exercise 1

Write a program with a function that swaps (exchanges the values of) two integer numbers. You should pass the arguments to the function by reference. Display the values of the two numbers before and after swapping them.

Exercise 1-Solution

```
#include <iostream>
using namespace std;
void swap(int& n1, int& n2) {
    int temp;
    temp = n1;
    n1 = n2;
    n2 = temp;
}
int main(){
    int a = 10, b = 20;
    cout << "Before swapping: ";
    cout << "a = " << a << ", b = " << b << endl;
    swap(a, b);
    cout << "After swapping: ";
    cout << "a = " << a << ", b = " << b << endl;
    return 0;
}
```

Exercise 2

Write a recursive function that computes the sum of the first 10 integer numbers (1+2+…+10 = 55)

Exercise 2-Solution

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

int sum(int n){
        if (n == 0)
                return n;
        else
                return n+sum(n-1);
}
int main(){
        int n = 10;
        cout << "Sum of first 10 numbers: " << sum(n) << endl;
        return 0;
}
```

Exercise 3

Write a function that takes a single integer argument (height) and displays a pyramid of this height made up of "*" characters on the screen. E.g.: when height=6 the following should be displayed on the screen.

```
Enter the height of the pyramid: 6
            *
          * * *
        * * * * *
      * * * * * * *
    * * * * * * * * *
  * * * * * * * * * * *
```

Exercise 3-Solution

```
#include <iostream>
using namespace std;
void pyramid(int height){
      int space = 0;
      for(int i = 1, k = 0; i <= height; ++i, k = 0){
            for(space = 1; space <= height-i; ++space)
                  cout <<"  ";
            while(k != 2*i-1){
                  cout << "* ";
                  ++k;}
            cout << endl;}}
int main(){
    int height = 0;
    cout <<"Enter the height of the pyramid: ";
    cin >> height;
    pyramid(height);
return 0;
}
```

Arrays

Why do we need arrays?

```
#include <iostream>

int main(){

    int a;
    int b;
    int c;

    return 0;
}
```

```
#include <iostream>

int main(){

    int a;
    int b;
    . . .
    int z;

    return 0;
}
```

Your program needs 3 `int` variables

Your program needs 1000 `int` variables!

Arrays

- A structure which stores many variables of the same type, e.g., `int, double, bool`, etc..
- Uses an ‘index’ to access each variable or ‘element’ of the array.
- C++ includes static arrays, allocated arrays, and standard library containers.

Array declaration

- Arrays are declared like normal variables.
- Square brackets [] indicate the number of variables which the array can store.

```
#include <iostream>

int main()
{
        int array[5];
        double array_d[2]={1.0,2.0};

        return 0;
}
```

array can store five `int` variables

array_d can store 2 `double` variables, initialized

Accessing array elements

- Array ‘index’ starts from zero
- Can be used for arithmetic, copied to other variables etc..

```
#include <iostream>

int main()
{
        int array[5];
        array[0] = 3;
        array[1] = array[0]+5;
        return 0;
}
```

array can store five `int` variables

set the first element to 3

set the second element to 8

Common memory errors

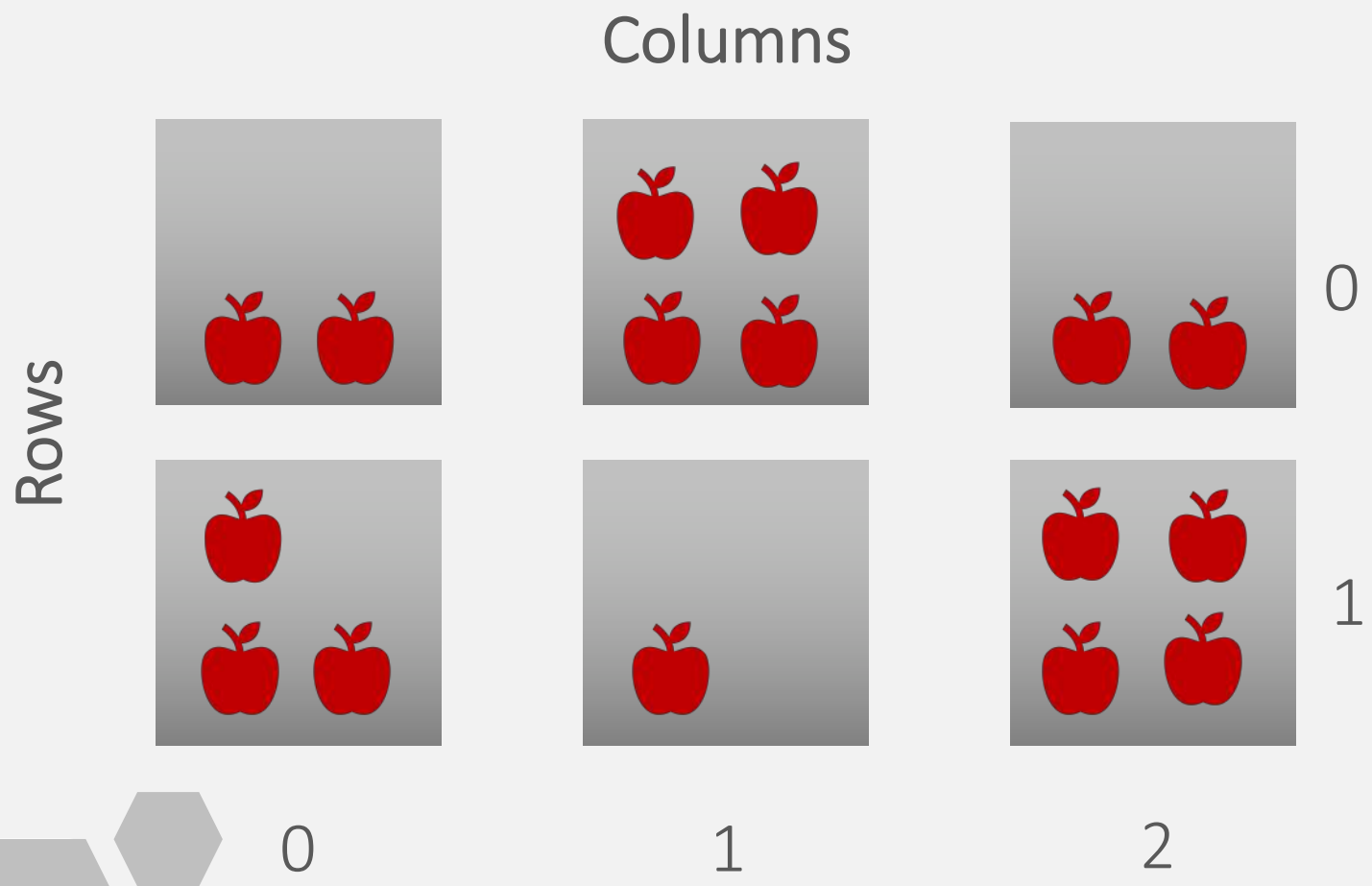
```
#include <iostream>

int main()
{
        int array[5];
        array[0] = 3;
        array[5] = 5;
        return 0;
}
```

array can store five `int` variables

set the first element to 3. OK

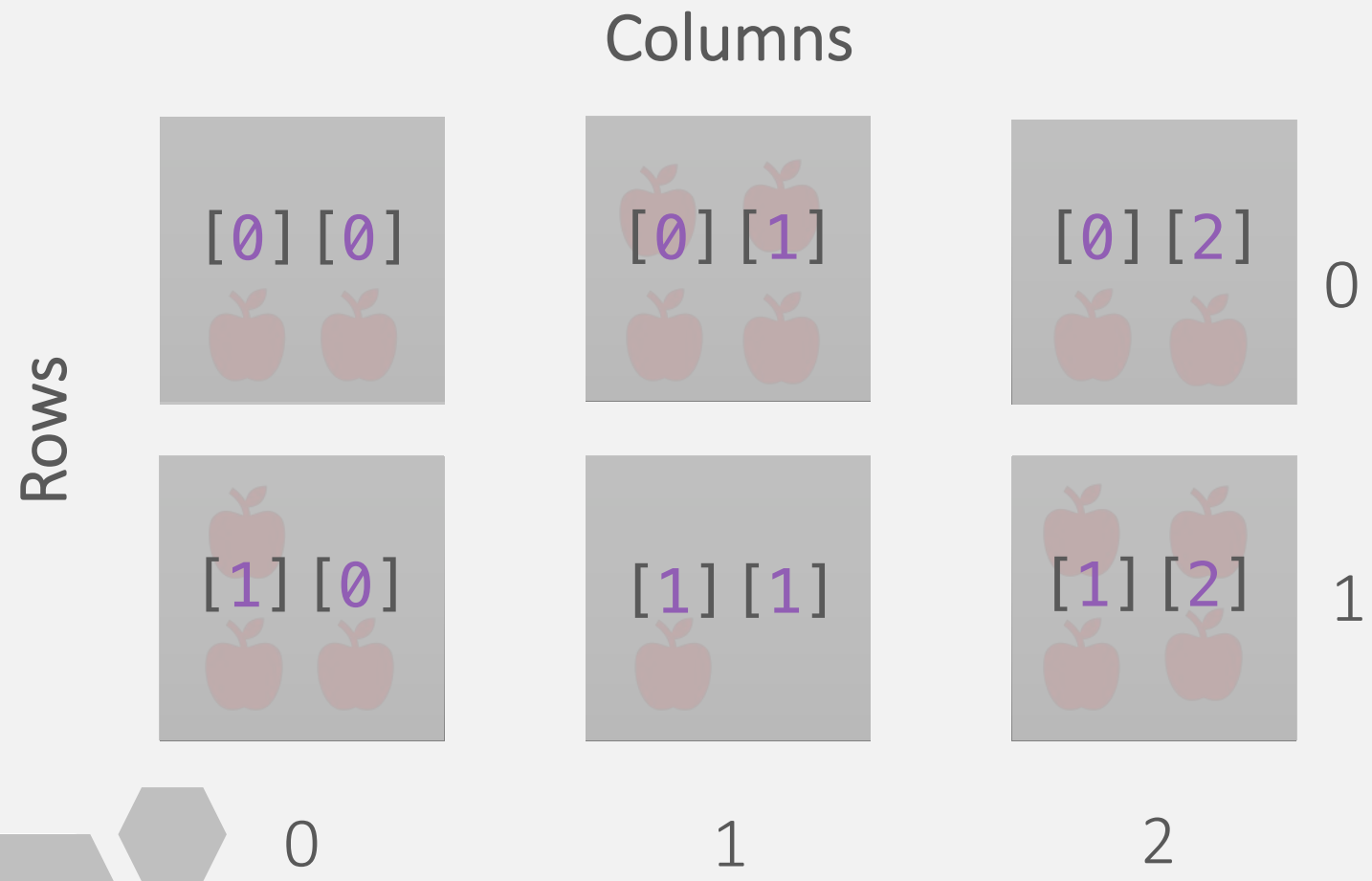
set the fifth element to 5. Oops!

Multidimensional Arrays




```
int array[2][3];
```

Multidimensional Arrays




```
int array[2][3];

array[0][0] = 2;
array[0][1] = 4;
array[0][2] = 2;
array[1][0] = 3;
array[1][1] = 1;
array[1][2] = 4;
```

Accessing Arrays using loops

```
#include <iostream>

int main()
{
        int array[5];

        for(int i=0; i<5; ++i)
            array[i] = 0;

        for(int i=0; i<5; ++i)
            array[i] += i;

        return 0;
}
```

loop over all elements and set to zero

Add `i` to each element of the array

Accessing 2D Arrays using loops

```
#include <iostream>

int main()
{
        int array[2][2];

        for(int i=0; i<2; ++i){
            for(int j=0; j<2; ++j){
                array[i][j] = 0;
            }
        }

        return 0;
}
```

array 2x2

Nested loop over all elements, set them to zero

Arrays and functions

- Arrays can be passed to a function as an argument.
- Only the name of the array is used as argument.
- When passing an array to a function, you need to mention the name of the array during the function call. Syntax:

```
function_name(array_name);
```

Exercise 1

Write a program that stores and calculates the sum of 3 integer numbers entered by the user (use arrays).

Exercise 1-Solution

```
#include <iostream>
using namespace std;
int main(){
      int num[3], sum = 0;
      cout << "Enter 3 numbers: " << endl;

      for(int i=0; i< 3; ++i){
            cin >> num[i];
            sum += num[i];
      }
      cout << "Sum is: " << sum << endl;

      return 0;
}
```

Exercise 2

Write a program that finds the smallest and largest element of a given array. E.g., if a[5]={6, 13, 8, 25, 100} then smallest number is a[0]=6 and largest number is a[4]=100.

Exercise 2-Solution

```
#include <iostream>
using namespace std;
int main(){
      int num[5]={6, 13, 8, 25, 100};
      int largest = 0, smallest = 100 ;

      for(int i=0; i< 5; ++i){
            if(num[i]>largest)
                largest = num[i];
            if(num[i]<smallest)
                smallest = num[i];
      }
      cout << "Smallest number is: " << smallest << endl;
      cout << "Largest number is: " << largest << endl;

      return 0;
}
```

Exercise 3

Write a program to reverse the elements of an array. Print the elements of the array before and after reversing them.

Exercise 3-Solution

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

void rvereseArray(int arr[], int start, int end) {
    if (start >= end)
        return;

    int temp = arr[start];
    arr[start] = arr[end];
    arr[end] = temp;
    rvereseArray(arr, start + 1, end - 1);
}
void printArray(int arr[], int size) {
   for (int i = 0; i < size; i++)
       cout << arr[i] << " ";
   cout << endl;
}
```

```cpp
int main() {
    int arr[] = {1, 2, 3, 4, 5};

    cout << "Original array is: " << endl;
    printArray(arr, 5);
    rvereseArray(arr, 0, 4);
    cout << "Reversed array is: " << endl;
    printArray(arr, 5);
    return 0;
}
```

Exercise 4

Write a program to implement the Tic Tac Toe game.

<u>Rules:</u>

-The game requires two players and it evolves on a 3x3 grid.

- Each player chooses to fill a square in the grid with a symbol assigned to him (X or O).

- In order to win, the players need to allign 3 identical symbols vertically, horizontally, or diagonally.

- If no one wins, then the game is said to be draw.

0	0	X
0	x	x
X	0	0

Exercise 4

Write a program to implement the Tic Tac Toe game.

<u>Hints:</u>

- Assign to a variable player the value 1 or 2 based on whether the number is even or odd. E.g.: player=(player%2)?1:2;

  This expression translates to :

```
If (player%2)
        player = 1;
else
        player = 2;
```

0	0	X
0	x	x
X	0	0

- Assign symbols based on whether the number of the player is even or odd.

  E.g.: mark = (player==1) ? 'X' : 'O' ;

Exercise 4-Solution

```cpp
#include <iostream>
using namespace std;
char square[10] = {'0','1','2','3','4','5','6','7','8','9'};
int checkwin();
void board();
int main(){
    int player = 1,i,choice;
    char mark;
    do{
        board();
        player=(player%2)?1:2;
        cout << "Player " << player << ", enter a number:  ";
        cin >> choice;
        mark=(player == 1) ? 'X' : 'O';
        if (choice == 1 && square[1] == '1')
            square[1] = mark;
        else if (choice == 2 && square[2] == '2')
            square[2] = mark;
        else if (choice == 3 && square[3] == '3')
            square[3] = mark;
        else if (choice == 4 && square[4] == '4')
            square[4] = mark;
        else if (choice == 5 && square[5] == '5')
            square[5] = mark;
        else if (choice == 6 && square[6] == '6')
            square[6] = mark;
        else if (choice == 7 && square[7] == '7')
            square[7] = mark;
        else if (choice == 8 && square[8] == '8')
            square[8] = mark;
        else if (choice == 9 && square[9] == '9')
            square[9] = mark;
        else{
            cout<<"Invalid move ";
            player--;
        }
        i=checkwin();
        player++;
    }while(i==-1);
    board();
    if(i==1)
        cout<<"==>\aPlayer "<<--player<<" win ";
    else
        cout<<"==>\aGame tie!";
    return 0;
}
```

Exercise 4-Solution(Cont.)

```cpp
/*****************************************
   FUNCTION TO RETURN GAME STATUS
    1 FOR GAME IS OVER WITH RESULT
    -1 FOR GAME IS IN PROGRESS
    0 GAME IS OVER AND NO RESULT
*****************************************/

int checkwin()
{
    if ((square[1] == square[2] && square[2] == square[3]) || (square[4] == square[5] && square[5] == square[6])
        || (square[7] == square[8] && square[8] == square[9]) || (square[1] == square[4] && square[4] == square[7])
        || (square[2] == square[5] && square[5] == square[8]) || (square[3] == square[6] && square[6] == square[9])
        || (square[1] == square[5] && square[5] == square[9]) || (square[3] == square[5] && square[5] == square[7]))
        return 1;
    else if (square[1] != '1' && square[2] != '2' && square[3] != '3'
            && square[4] != '4' && square[5] != '5' && square[6] != '6'
            && square[7] != '7' && square[8] != '8' && square[9] != '9')
        return 0;
    else
        return -1;
}
/*****************************************************************
     FUNCTION TO DRAW BOARD OF TIC TAC TOE WITH PLAYERS MARK
*****************************************************************/
void board(){
    cout << "Player 1 (X)  -  Player 2 (O)" << endl;
    cout << "     |     |     " << endl;
    cout << "  " << square[1] << "  |  " << square[2] << "  |  " << square[3] << endl;
    cout << "_____|_____|_____" << endl;
    cout << "     |     |     " << endl;
    cout << "  " << square[4] << "  |  " << square[5] << "  |  " << square[6] << endl;
    cout << "_____|_____|_____" << endl;
    cout << "     |     |     " << endl;
    cout << "  " << square[7] << "  |  " << square[8] << "  |  " << square[9] << endl;
    cout << "     |     |     " << endl << endl;
}
```

Additional Resources

- http://www.cplusplus.com/doc/tutorial/
- https://en.cppreference.com/w/
- Programming: Principles and Practice Using C++, Bjarne Stroustrup (Updated for C++11/C++14)
- C++ Primer, Stanley Lippman, Josée Lajoie, and Barbara E. Moo (Updated for C++11)