C/C++ Program Design

LAB 3

CONTENTS

- Learn how to create and use arrays(Declare, Initialize and Access)
- Master character arrays and strings
- Learn how to create and use structures(Declare, Initialize and Access)

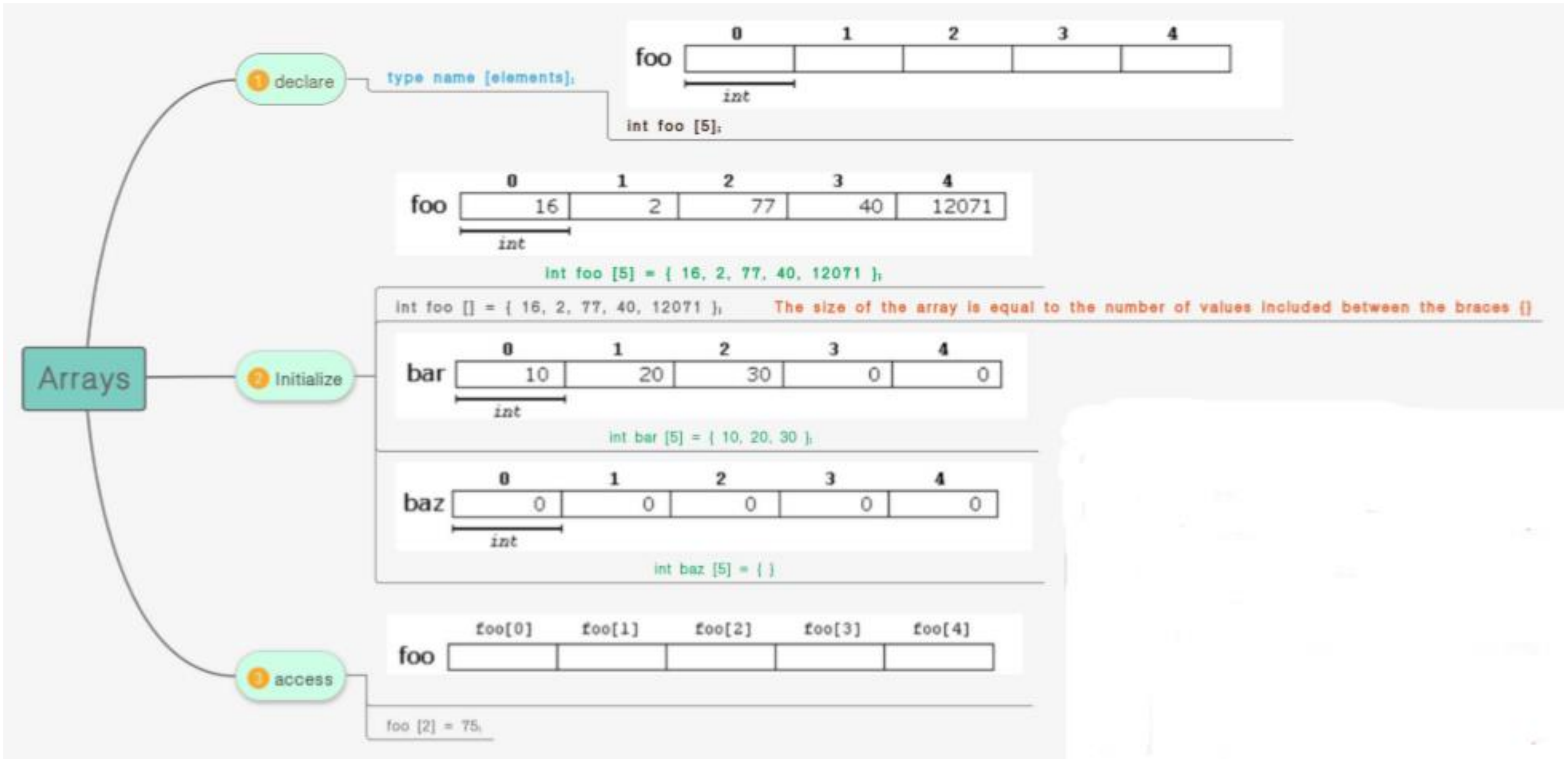
2 Knowledge Points

2.1 Array

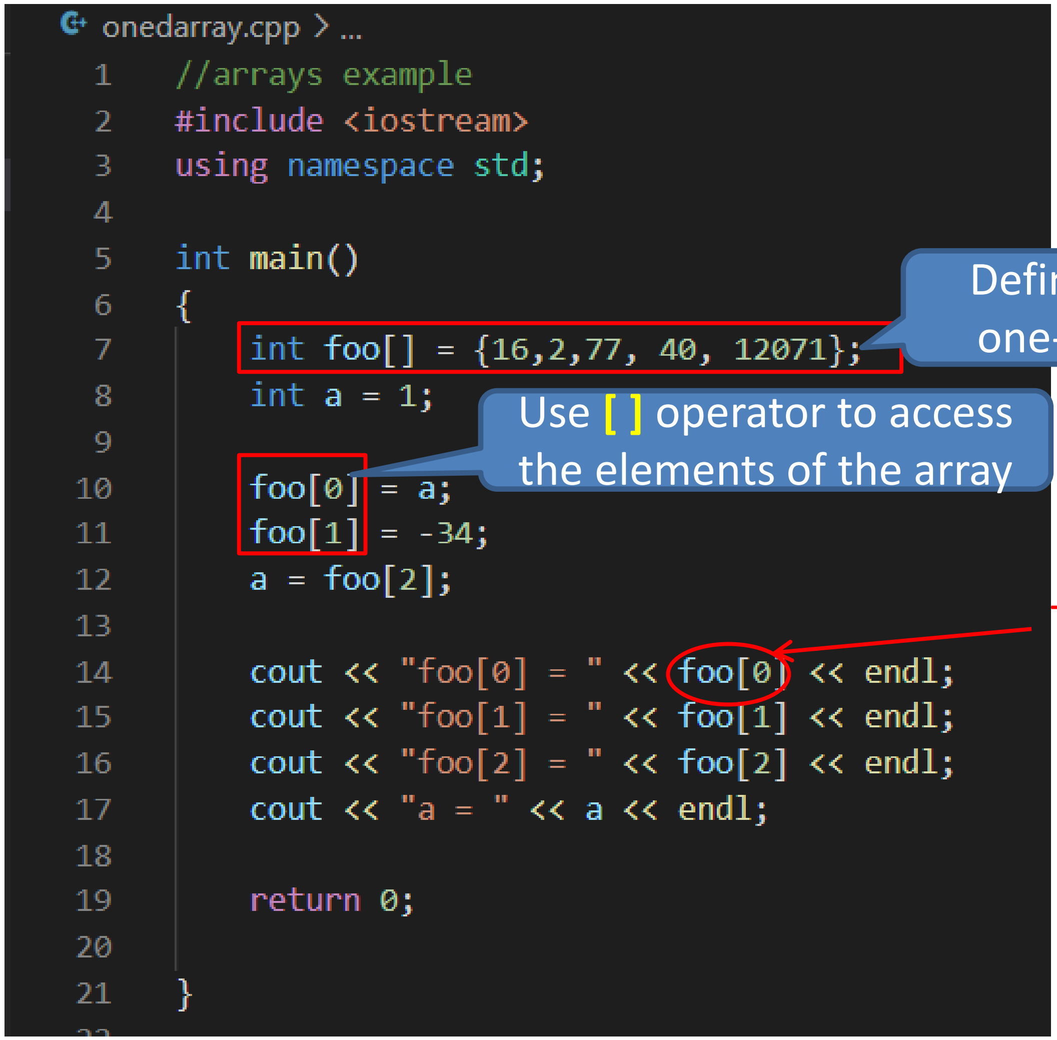
2.2 Character arrays and strings

2.3 Structure

2.1 Array



Arrays
1 declare
type name [elements];
foo
0 1 2 3 4
int
int foo [5];
2 Initialize
foo 16 2 77 40 12071
int foo [5] = { 16, 2, 77, 40, 12071 };
int foo [] = { 16, 2, 77, 40, 12071 };
The size of the array is equal to the number of values included between the braces {}
bar 10 20 30 0 0
int bar [5] = { 10, 20, 30 };
baz 0 0 0 0 0
int baz [5] = { }
3 access
foo[0] foo[1] foo[2] foo[3] foo[4]
foo [2] = 75;

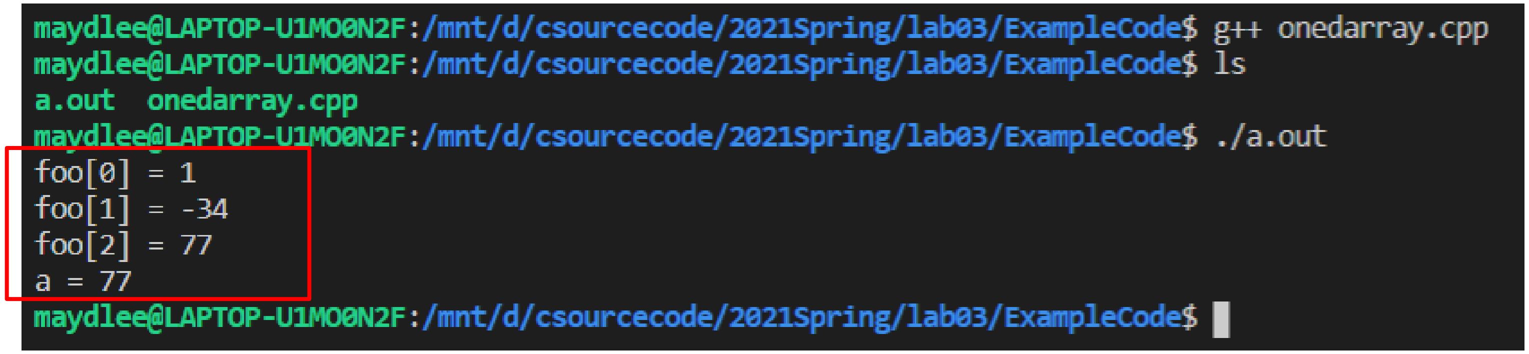


```
onedarray.cpp > ...
//arrays example
#include <iostream>
using namespace std;

int main()
{
    int foo[] = {16,2,77, 40, 12071};
    int a = 1;

    foo[0] = a;
    foo[1] = -34;
    a = foo[2];

    cout << "foo[0] = " << foo[0] << endl;
    cout << "foo[1] = " << foo[1] << endl;
    cout << "foo[2] = " << foo[2] << endl;
    cout << "a = " << a << endl;

    return 0;

}
```

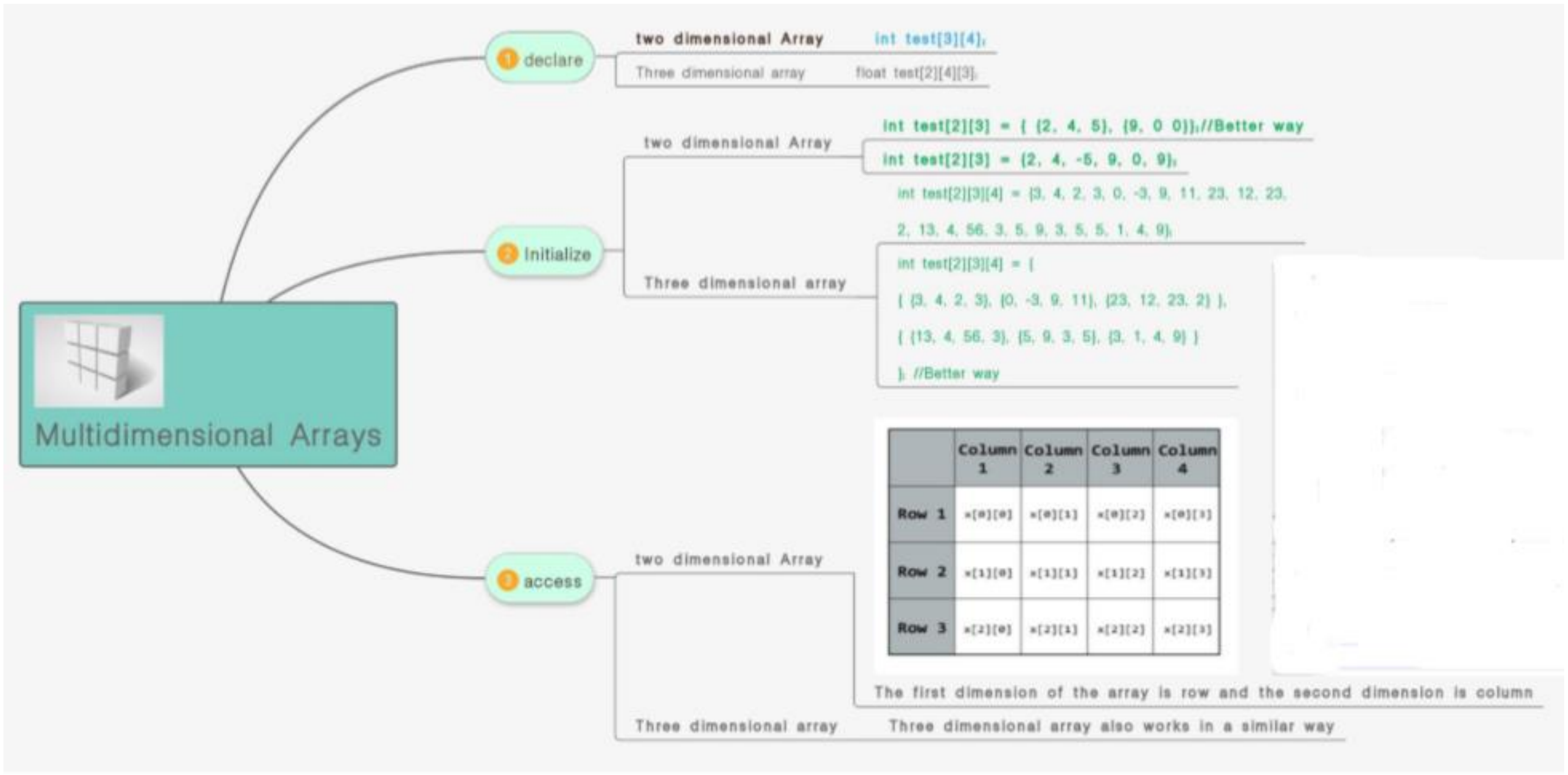
Define and initialize a one-dimension array

Use [ ] operator to access the elements of the array

The array index starts from 0



```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ g++ onedarray.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ls
a.out  onedarray.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ./a.out
foo[0] = 1
foo[1] = -34
foo[2] = 77
a = 77
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$
```



Multidimensional Arrays

1 declare

two dimensional Array
```
int test[3][4];
```

Three dimensional array
```
float test[2][4][3];
```

2 Initialize

two dimensional Array
```
int test[2][3] = { {2, 4, 5}, {9, 0 0}};//Better way
int test[2][3] = {2, 4, -5, 9, 0, 9};
```

Three dimensional array
```
int test[2][3][4] = {3, 4, 2, 3, 0, -3, 9, 11, 23, 12, 23,
2, 13, 4, 56, 3, 5, 9, 3, 5, 5, 1, 4, 9};
```
```
int test[2][3][4] = {
{ {3, 4, 2, 3}, {0, -3, 9, 11}, {23, 12, 23, 2} },
{ {13, 4, 56, 3}, {5, 9, 3, 5}, {3, 1, 4, 9} }
}; //Better way
```

3 access

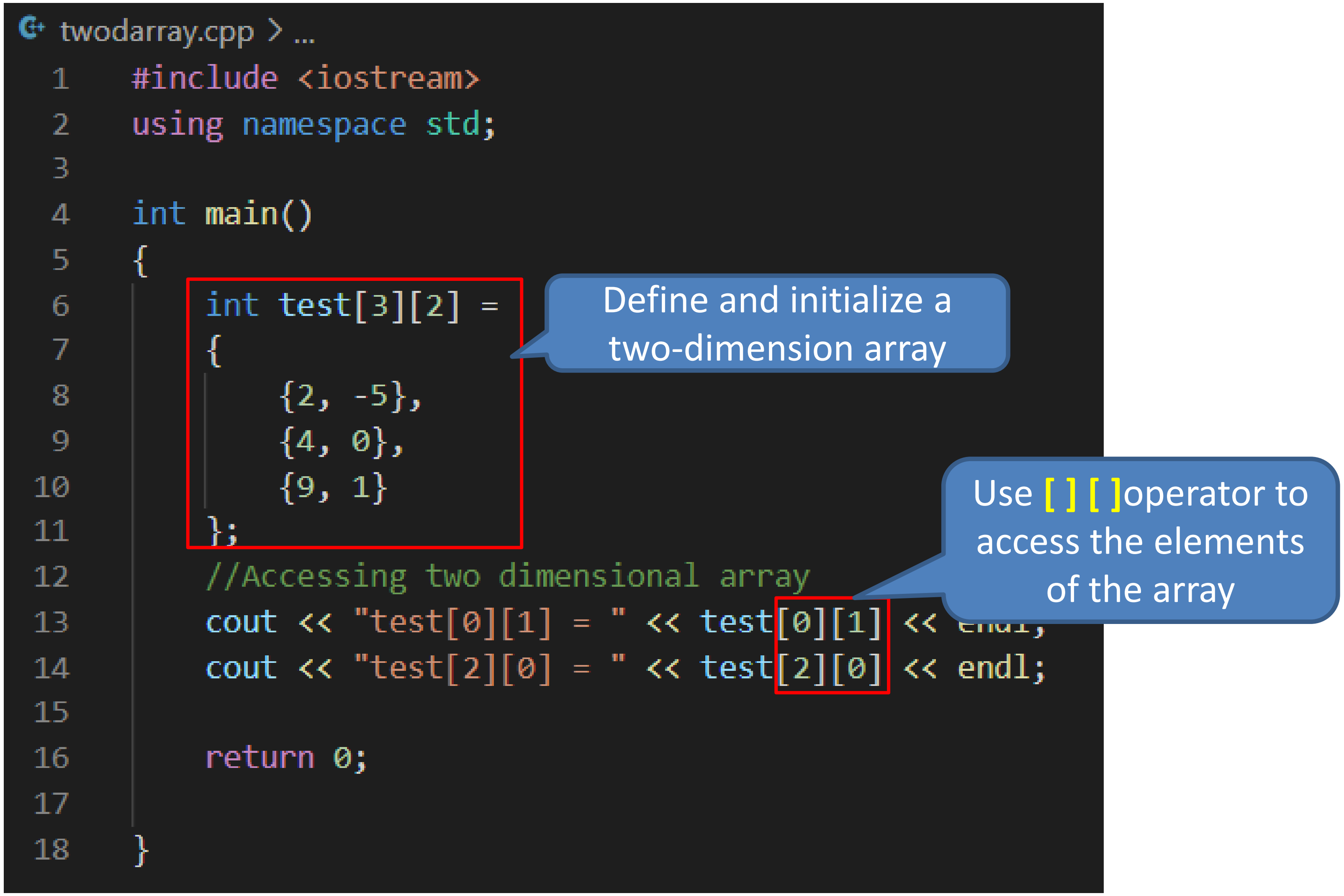
two dimensional Array

	Column 1	Column 2	Column 3	Column 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

The first dimension of the array is row and the second dimension is column

Three dimensional array

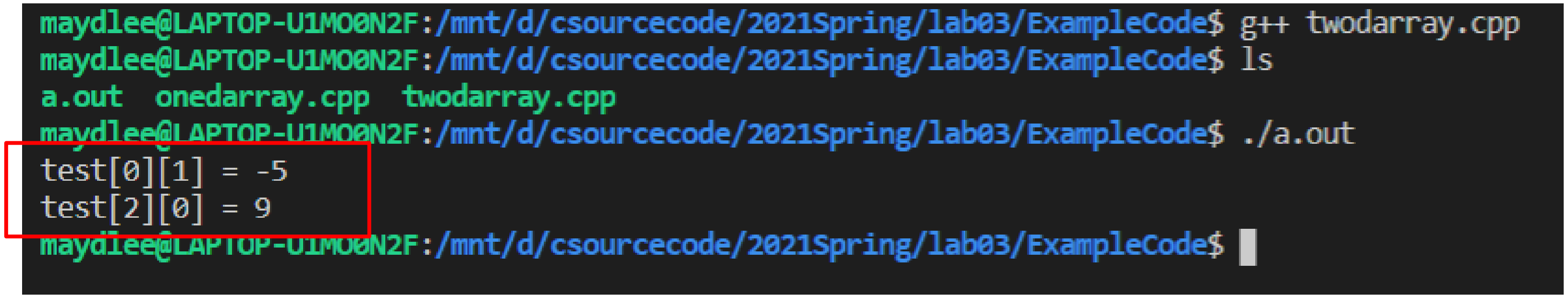
Three dimensional array also works in a similar way



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

int main()
{
    int test[3][2] =
    {
        {2, -5},
        {4, 0},
        {9, 1}
    };
    //Accessing two dimensional array
    cout << "test[0][1] = " << test[0][1] << endl;
    cout << "test[2][0] = " << test[2][0] << endl;

    return 0;

}
```

Define and initialize a two-dimension array

Use [ ] [ ]operator to access the elements of the array



```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ g++ twodarray.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ls
a.out  onedarray.cpp  twodarray.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ./a.out
test[0][1] = -5
test[2][0] = 9
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ 
```

2.2 Character array and strings

2.2.1 Define a C-string

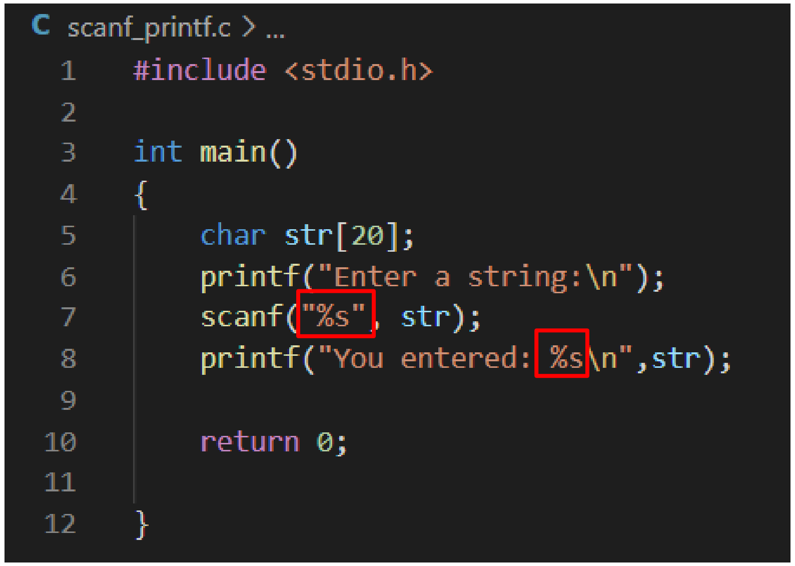
You can use one of the four ways below to define a character array:

```
char str[ ] = “C++”  ;
char str[4] = “C++”  ;
char str[ ] = {‘C’, ‘+’, ‘+’, ‘\0’};
char str[4] = {‘C’, ‘+’, ‘+’, ‘\0’}
```

Strings end with \0

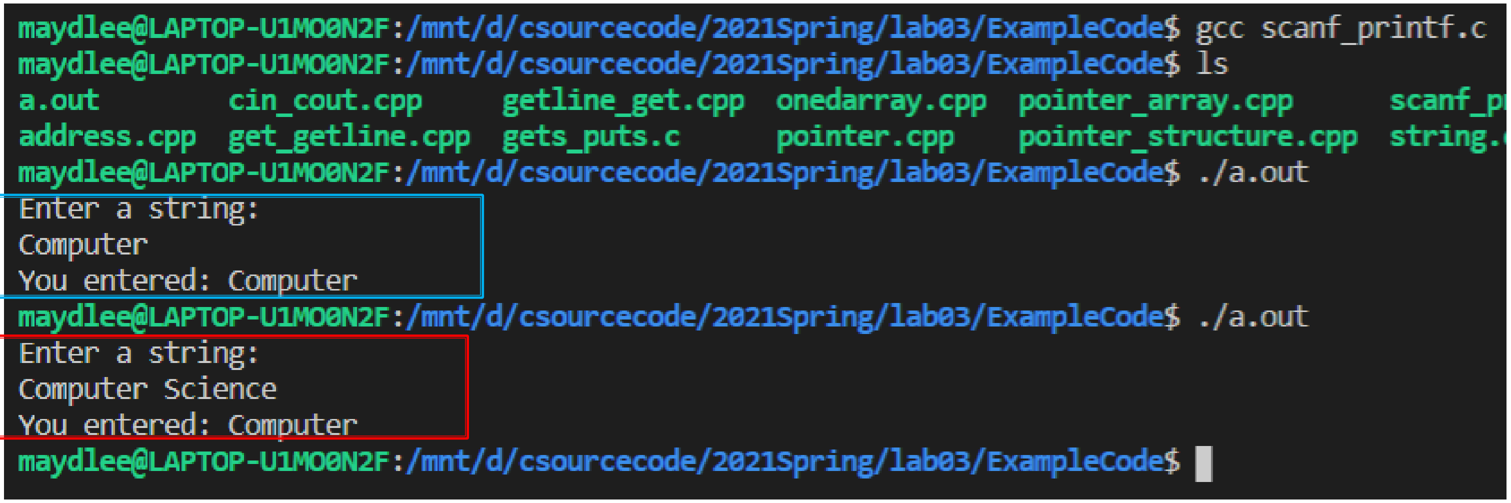
2.2.2 Keyboard input and terminal output of character array

1. C: scanf & printf
   - %d ----int
   - %f ----float
   - %c -----char
   - %s -----string



scanf_printf.c > ...

```c
#include <stdio.h>

int main()
{
    char str[20];
    printf("Enter a string:\n");
    scanf("%s", str);
    printf("You entered: %s\n",str);

    return 0;

}
```



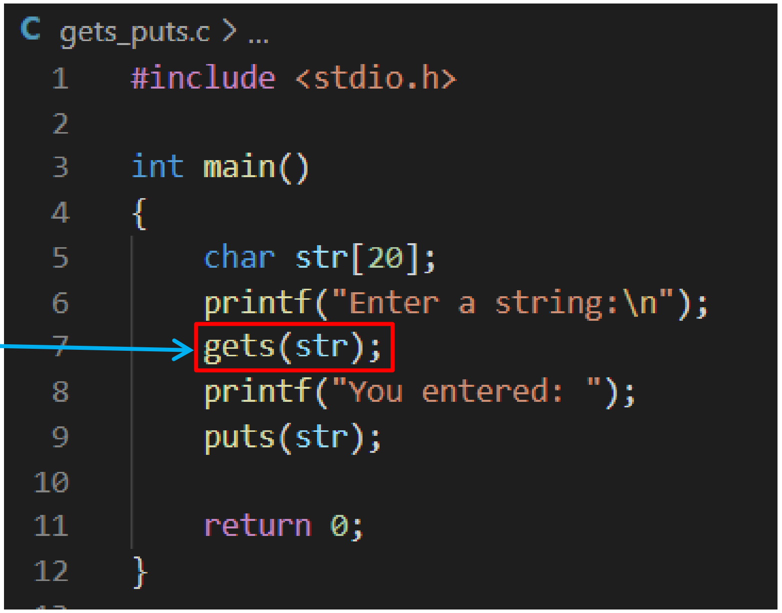
```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ gcc scanf_printf.c
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ls
a.out         cin_cout.cpp     getline_get.cpp  onedarray.cpp  pointer_array.cpp      scanf_pr
address.cpp   get_getline.cpp  gets_puts.c      pointer.cpp    pointer_structure.cpp  string.
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ./a.out
Enter a string:
Computer
You entered: Computer
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ./a.out
Enter a string:
Computer Science
You entered: Computer
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$
```

Why only Computer?

scanf uses whitespace—spaces, tabs, and newlines to delineate a string.

2.2.2 Keyboard input and terminal output of Character array

2. C: gets & puts

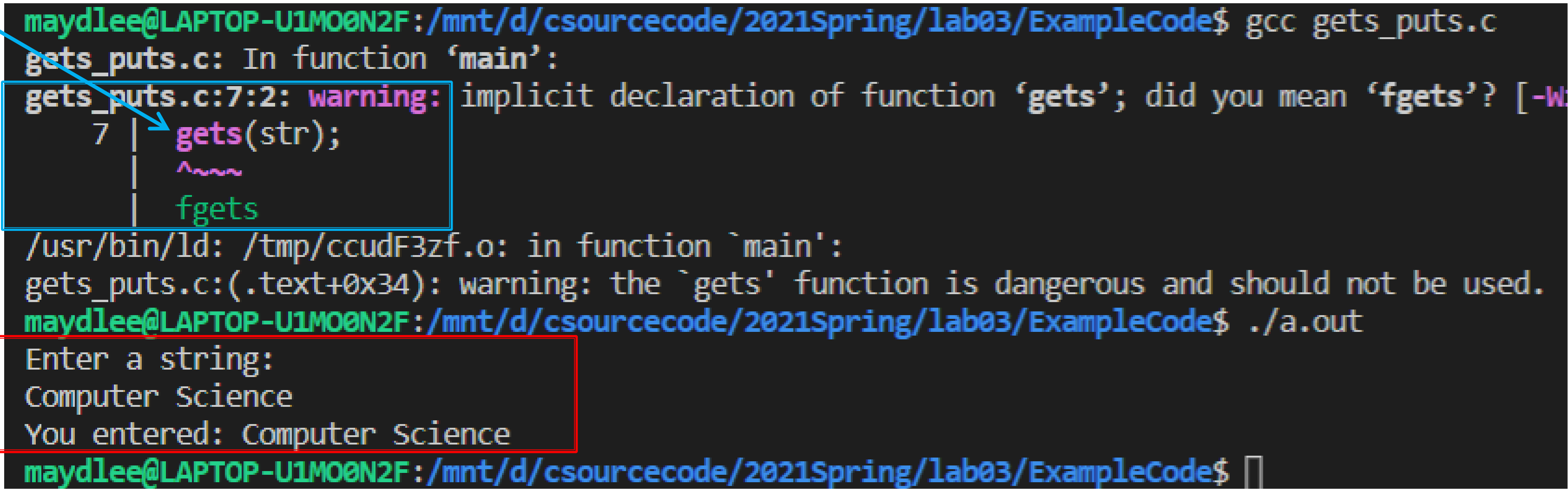


```
gets_puts.c > ...
1   #include <stdio.h>
2
3   int main()
4   {
5       char str[20];
6       printf("Enter a string:\n");
7       gets(str);
8       printf("You entered: ");
9       puts(str);
10
11      return 0;
12  }
```

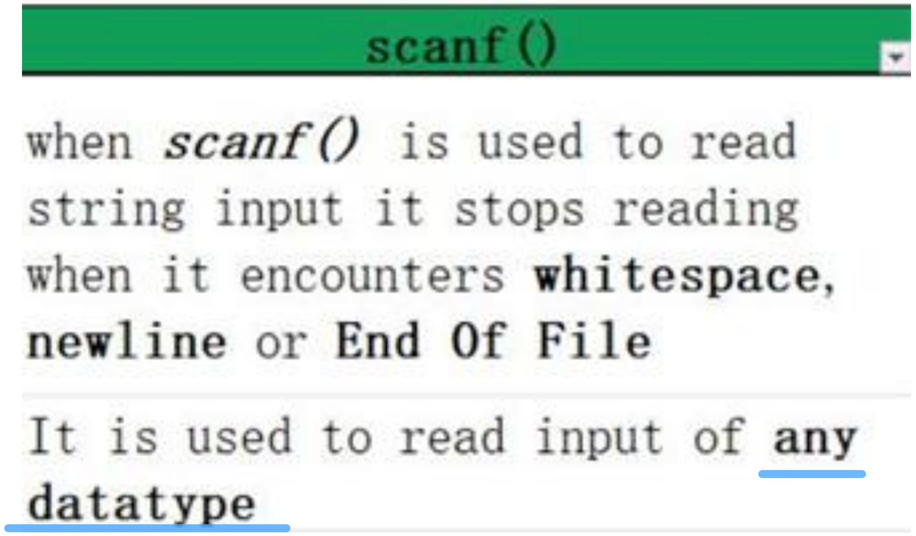
```
fgets(str, 20, stdin);
```

There is a warning due to using gets().
You can use fgets() function instead.

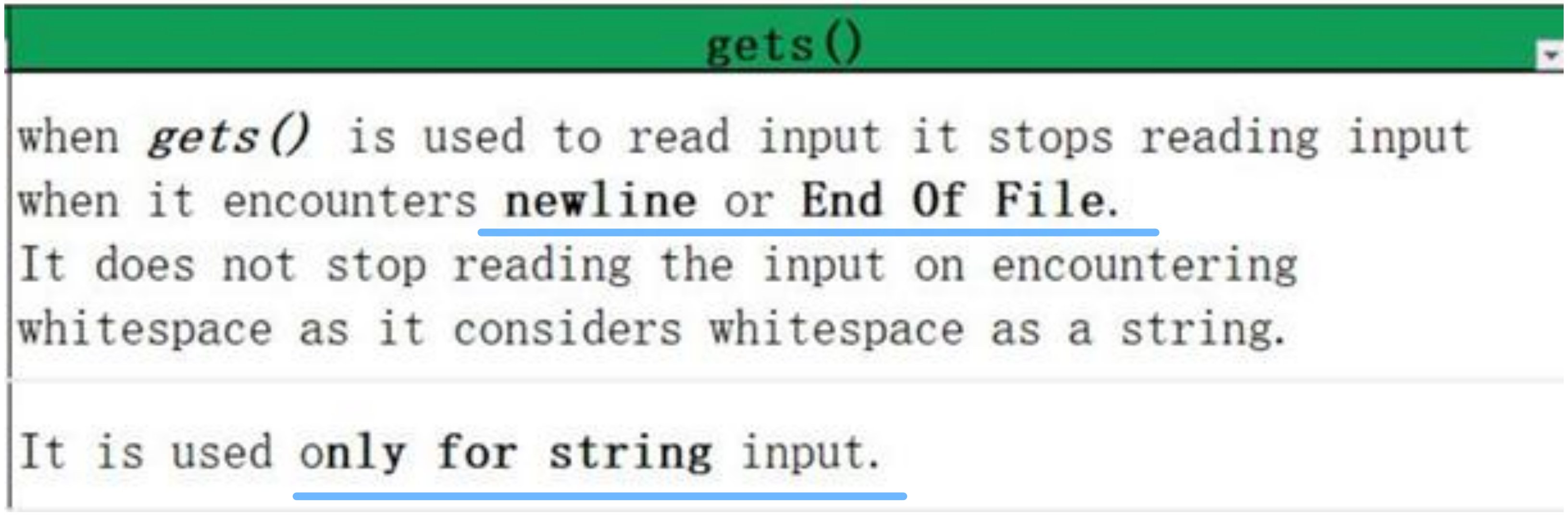
Use gets to gain the whole line



```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ gcc gets_puts.c
gets_puts.c: In function ‘main’:
gets_puts.c:7:2: warning: implicit declaration of function ‘gets’; did you mean ‘fgets’? [-W
    7 |  gets(str);
      |  ^~~~
      |  fgets
/usr/bin/ld: /tmp/ccudF3zf.o: in function `main':
gets_puts.c:(.text+0x34): warning: the `gets' function is dangerous and should not be used.
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ./a.out
Enter a string:
Computer Science
You entered: Computer Science
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ 
```



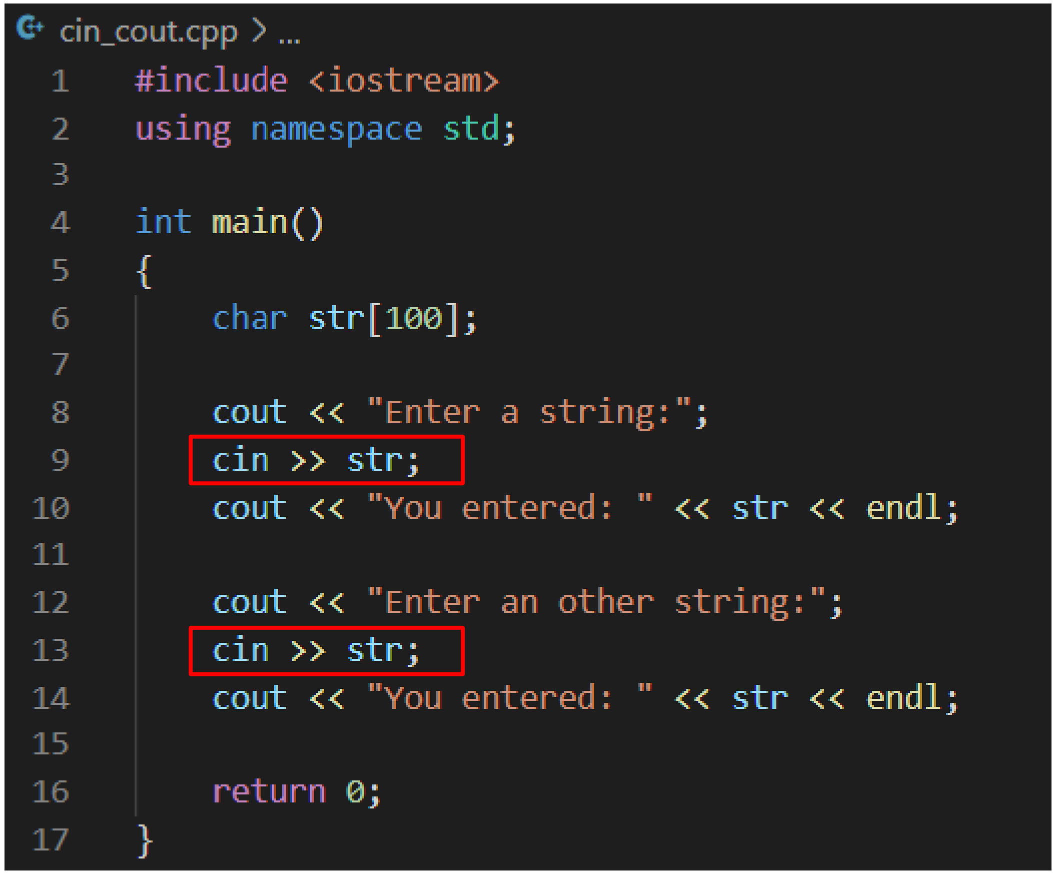
scanf()
when *scanf()* is used to read string input it stops reading when it encounters **whitespace**, **newline** or **End Of File**
It is used to read input of **any datatype**



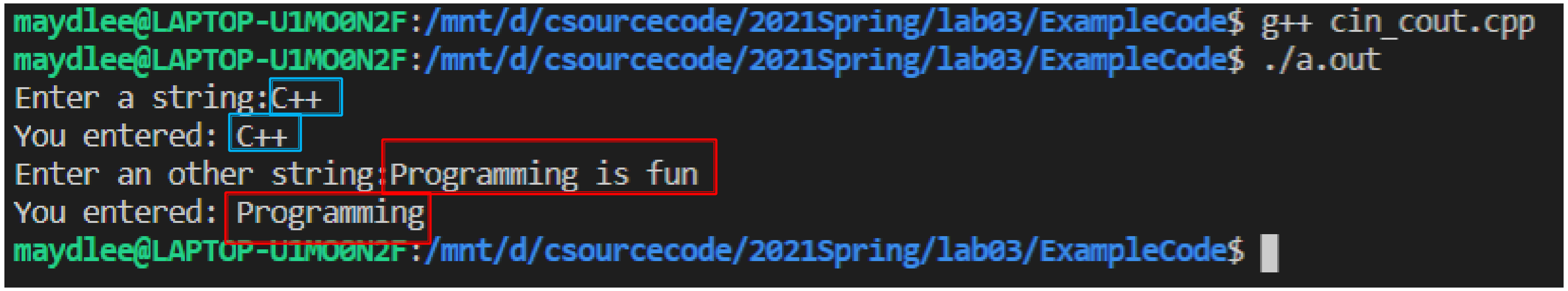
gets()
when *gets()* is used to read input it stops reading input when it encounters **newline** or **End Of File**. It does not stop reading the input on encountering whitespace as it considers whitespace as a string.
It is used **only for string** input.

2.2.2 Keyboard input and terminal output of Character array

3. C++: cin & cout



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

int main()
{
    char str[100];

    cout << "Enter a string:";
    cin >> str;
    cout << "You entered: " << str << endl;

    cout << "Enter an other string:";
    cin >> str;
    cout << "You entered: " << str << endl;

    return 0;
}
```

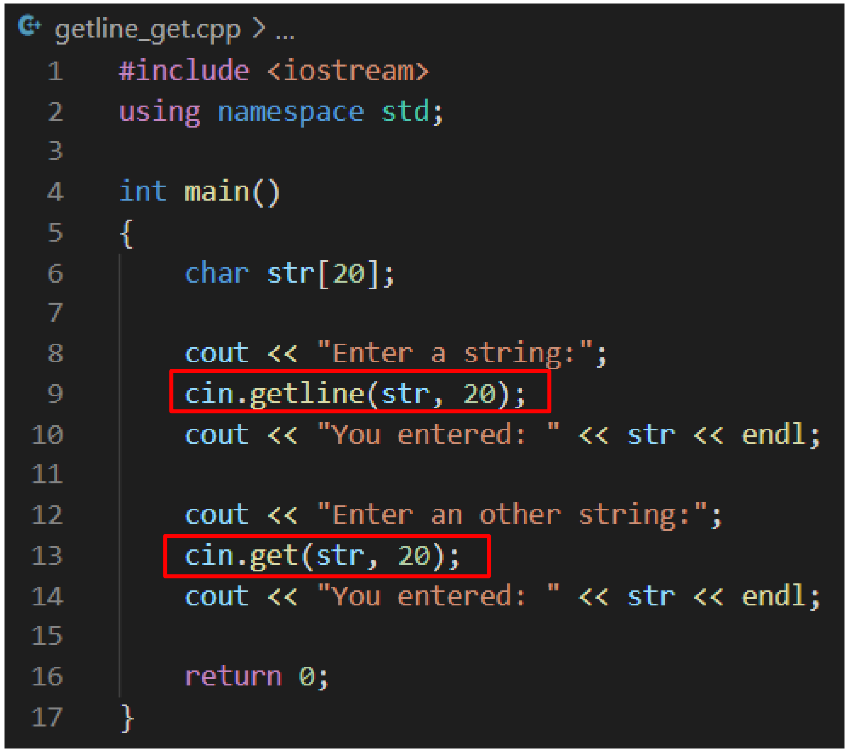


```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ g++ cin_cout.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ./a.out
Enter a string:C++
You entered: C++
Enter an other string:Programming is fun
You entered: Programming
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$
```

The cin is to use whitespace-- spaces, tabs, and newlines to delineate a string.

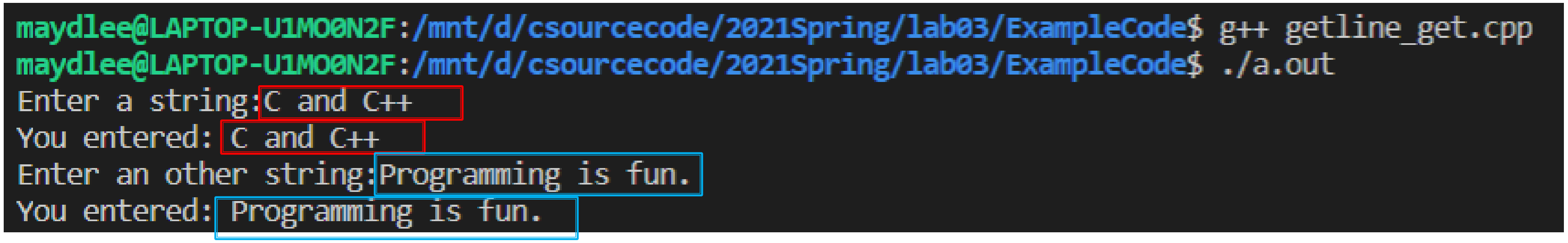
2.2.2 Keyboard input and terminal output of Character array

4. C++: cin.getline( ) & cin.get( )



getline_get.cpp > ...

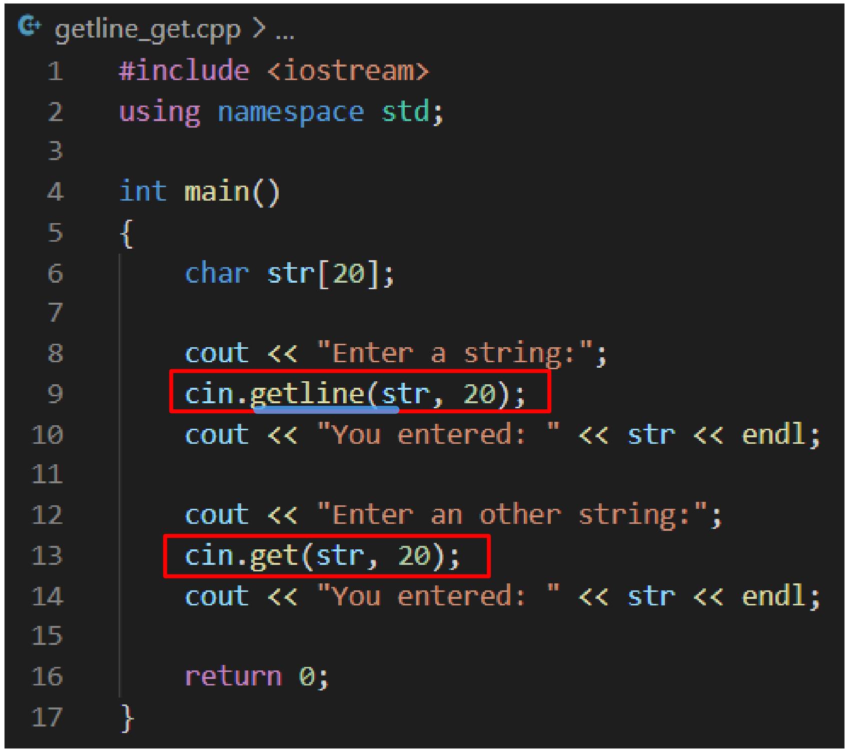
```cpp
#include <iostream>
using namespace std;

int main()
{
    char str[20];

    cout << "Enter a string:";
    cin.getline(str, 20);
    cout << "You entered: " << str << endl;

    cout << "Enter an other string:";
    cin.get(str, 20);
    cout << "You entered: " << str << endl;

    return 0;
}
```



```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ g++ getline_get.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ./a.out
Enter a string:C and C++
You entered: C and C++
Enter an other string:Programming is fun.
You entered: Programming is fun.
```

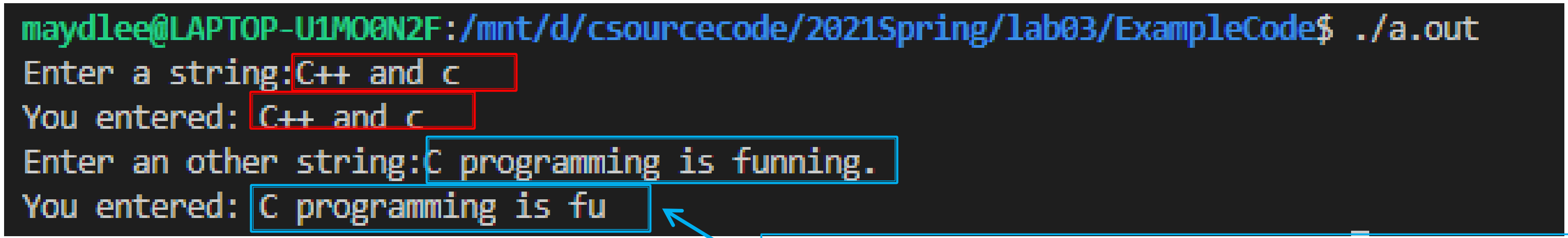
2.2.2 Keyboard input and terminal output of Character array

4. C++: cin.getline( ) & cin.get( )



getline_get.cpp > ...

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

int main()
{
    char str[20];

    cout << "Enter a string:";
    cin.getline(str, 20);
    cout << "You entered: " << str << endl;

    cout << "Enter an other string:";
    cin.get(str, 20);
    cout << "You entered: " << str << endl;

    return 0;
}
```

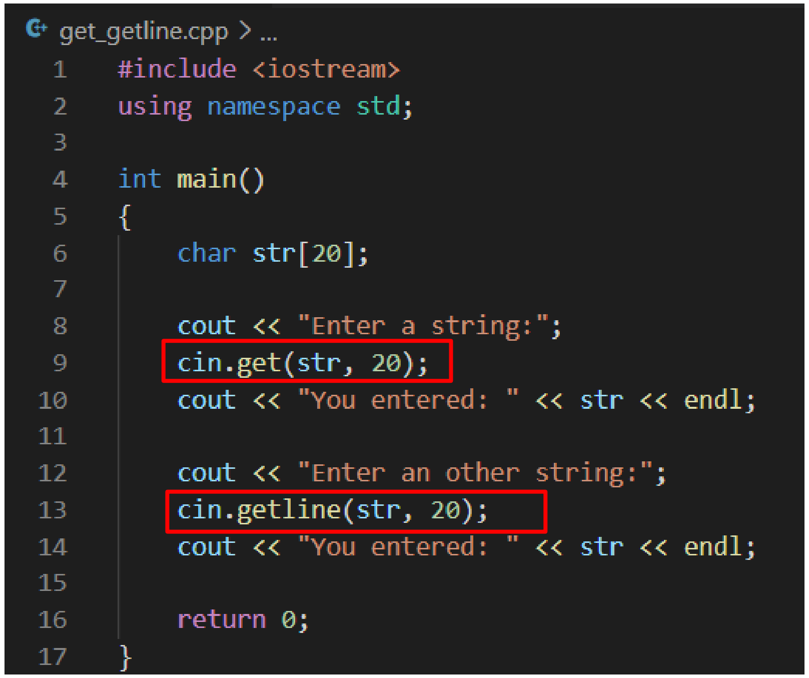


```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ./a.out
Enter a string:C++ and c
You entered: C++ and c
Enter an other string:C programming is funning.
You entered: C programming is fu
```

If the length of input string is greater than 20, it can only store first 19 characters in str.

2.2.2 Keyboard input and terminal output of Character array

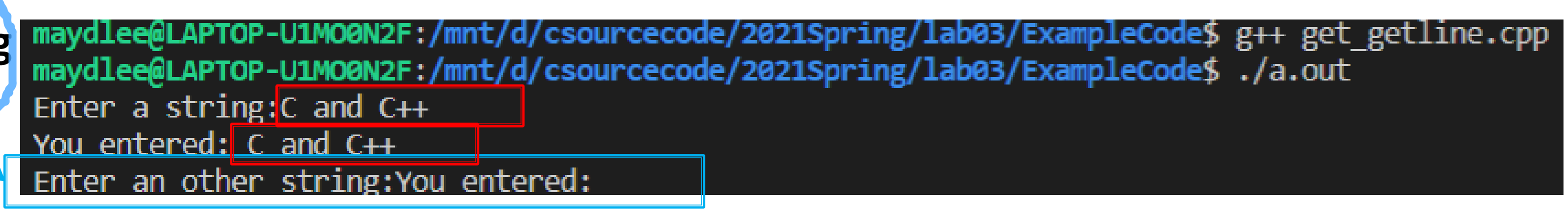
4. C++: cin.get( ) & cin.getline( )



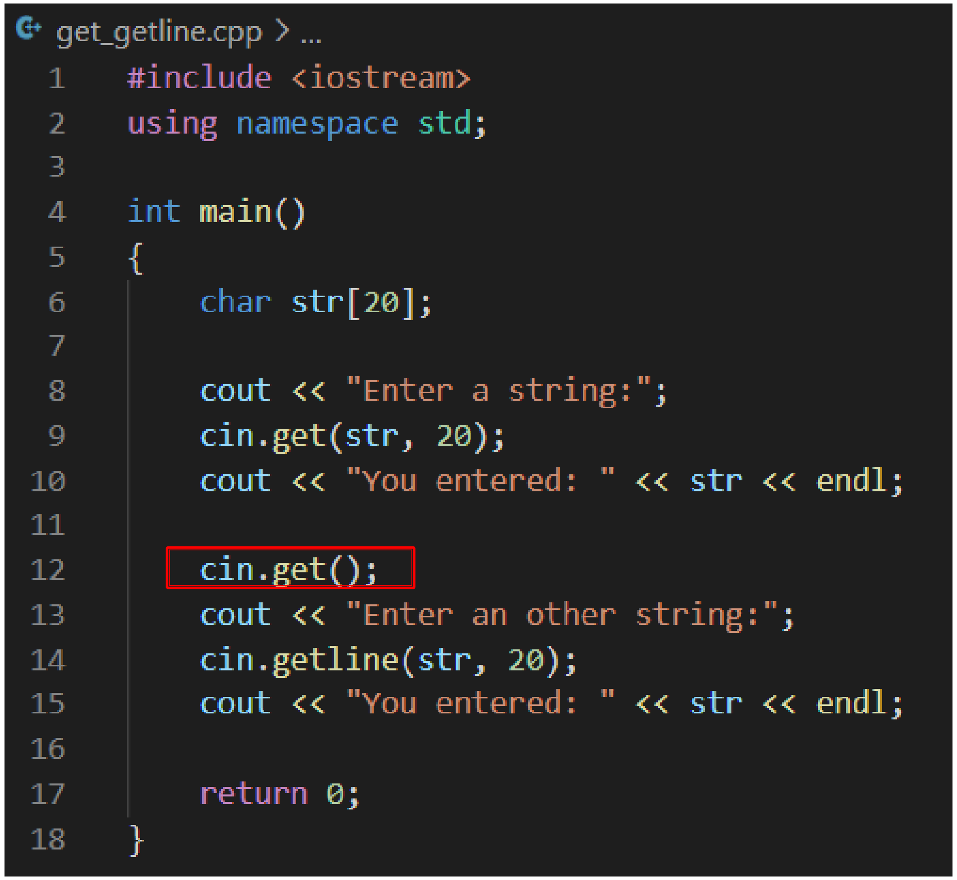
```
get_getline.cpp > ...
#include <iostream>
using namespace std;

int main()
{
    char str[20];

    cout << "Enter a string:";
    cin.get(str, 20);
    cout << "You entered: " << str << endl;

    cout << "Enter an other string:";
    cin.getline(str, 20);
    cout << "You entered: " << str << endl;

    return 0;
}
```

getline() and get() both read an entire input line—that is, up until a newline character. However, getline() discard the newline character, whereas get() leave it in the input queue.

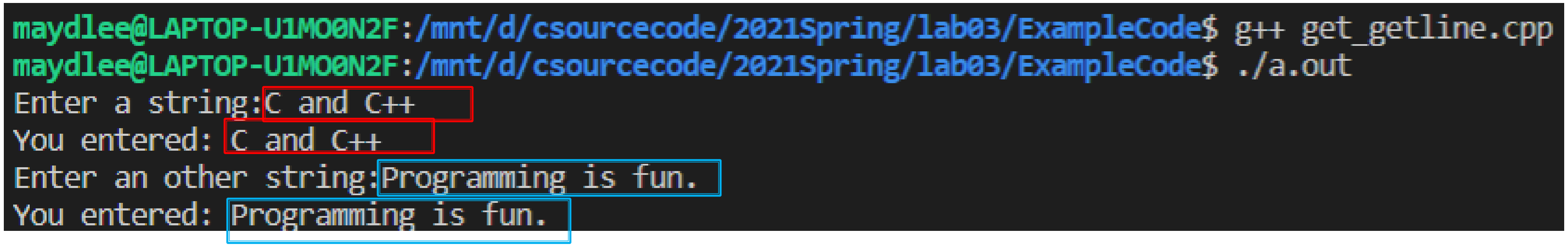
Program runs without entering another string



```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ g++ get_getline.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ./a.out
Enter a string:C and C++
You entered: C and C++
Enter an other string:You entered:
```



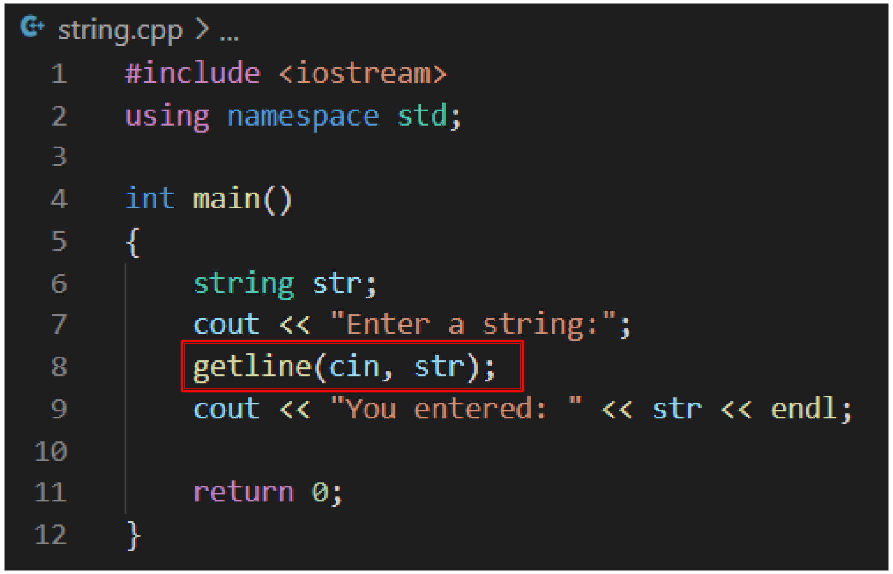
```
get_getline.cpp > ...
#include <iostream>
using namespace std;

int main()
{
    char str[20];

    cout << "Enter a string:";
    cin.get(str, 20);
    cout << "You entered: " << str << endl;

    cin.get();
    cout << "Enter an other string:";
    cin.getline(str, 20);
    cout << "You entered: " << str << endl;

    return 0;
}
```



```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ g++ get_getline.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ./a.out
Enter a string:C and C++
You entered: C and C++
Enter an other string:Programming is fun.
You entered: Programming is fun.
```

2.2.3 Keyboard input and terminal output of C++ string

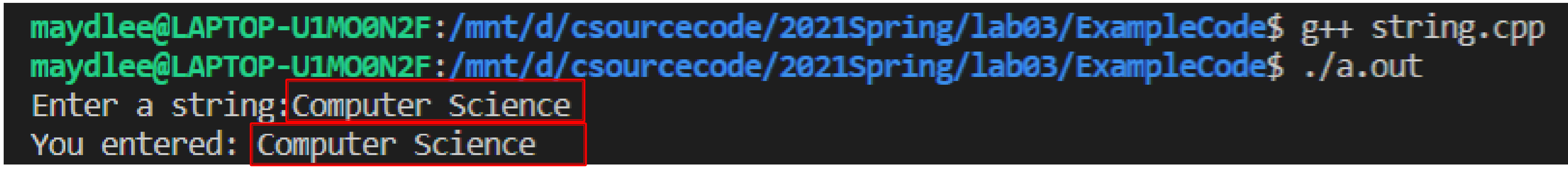
C++ string using string data type



string.cpp > ...

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

int main()
{
    string str;
    cout << "Enter a string:";
    getline(cin, str);
    cout << "You entered: " << str << endl;

    return 0;
}
```

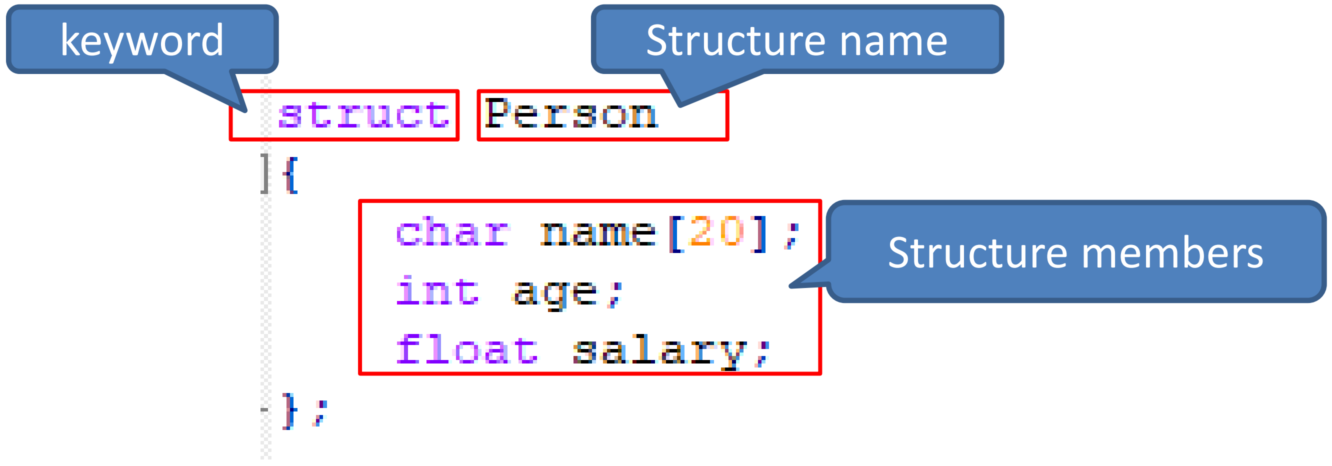
getline() function takes the input stream as the first parameter which is cin and str as the location of the line to be stored.



```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ g++ string.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ./a.out
Enter a string:Computer Science
You entered: Computer Science
```

2.3 Structure

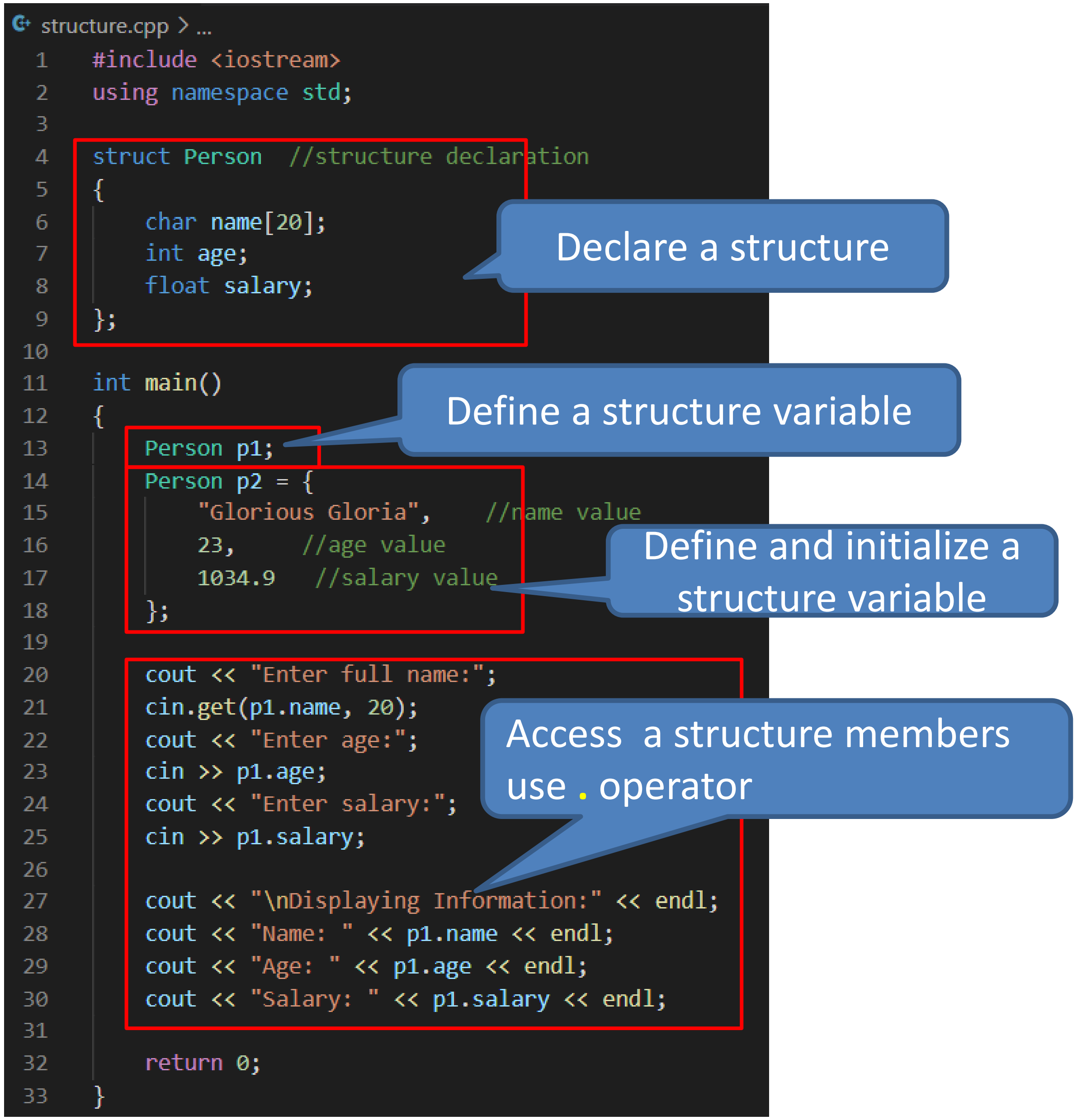
2.3.1 Declare a structure



```c
struct Person
{
        char name[20];
        int age;
        float salary;
};
```

When a structure is declared, no memory is allocated.

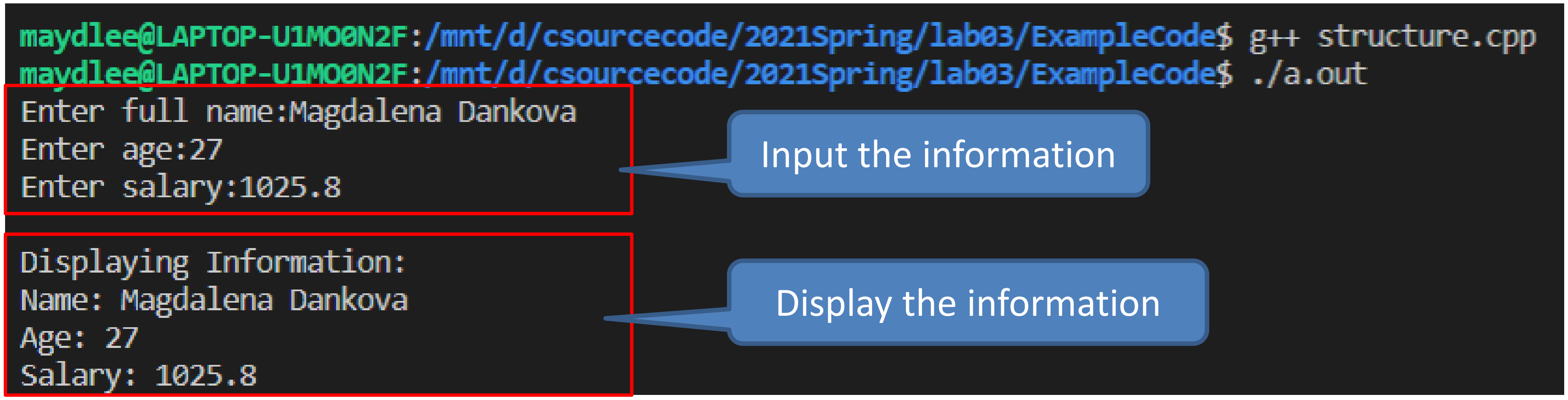
2.3.2 Define,initialize and access a structure variable



structure.cpp > ...

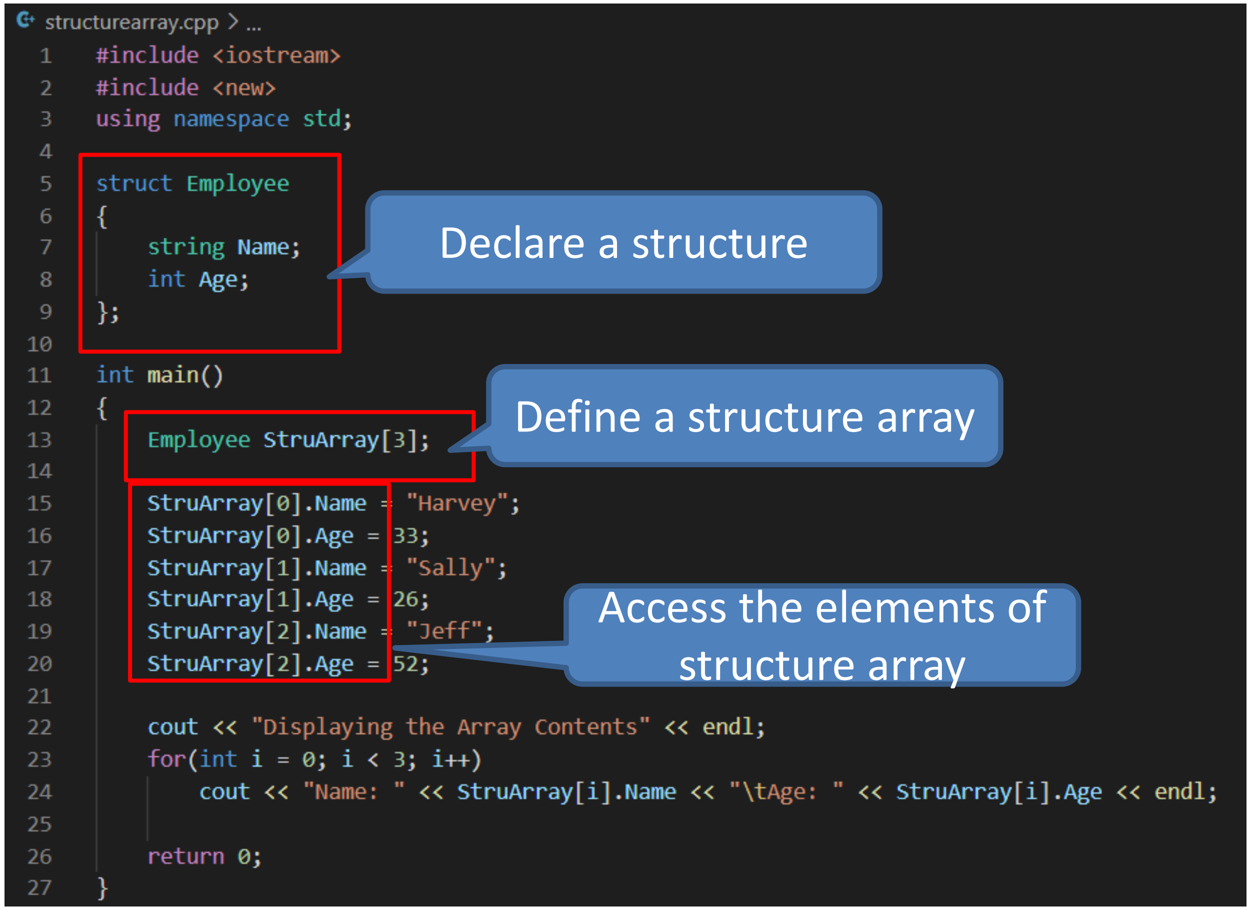
```cpp
#include <iostream>
using namespace std;

struct Person  //structure declaration
{
    char name[20];
    int age;
    float salary;
};

int main()
{
    Person p1;
    Person p2 = {
        "Glorious Gloria",    //name value
        23,     //age value
        1034.9   //salary value
    };

    cout << "Enter full name:";
    cin.get(p1.name, 20);
    cout << "Enter age:";
    cin >> p1.age;
    cout << "Enter salary:";
    cin >> p1.salary;

    cout << "\nDisplaying Information:" << endl;
    cout << "Name: " << p1.name << endl;
    cout << "Age: " << p1.age << endl;
    cout << "Salary: " << p1.salary << endl;

    return 0;
}
```

Declare a structure
Define a structure variable
Define and initialize a structure variable
Access a structure members use . operator



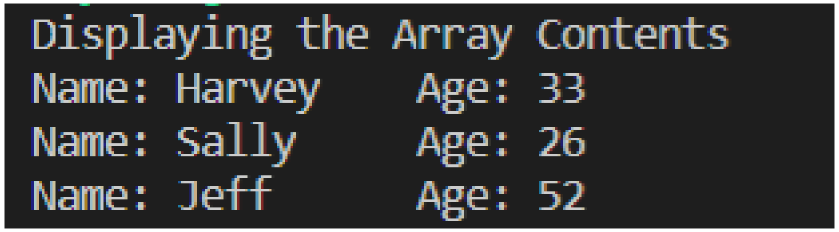
```
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ g++ structure.cpp
maydlee@LAPTOP-U1MO0N2F:/mnt/d/csourcecode/2021Spring/lab03/ExampleCode$ ./a.out
Enter full name:Magdalena Dankova
Enter age:27
Enter salary:1025.8

Displaying Information:
Name: Magdalena Dankova
Age: 27
Salary: 1025.8
```

2.3.3 Array of Structure



structurearray.cpp > ...

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

struct Employee
{
    string Name;
    int Age;
};

int main()
{
    Employee StruArray[3];

    StruArray[0].Name = "Harvey";
    StruArray[0].Age = 33;
    StruArray[1].Name = "Sally";
    StruArray[1].Age = 26;
    StruArray[2].Name = "Jeff";
    StruArray[2].Age = 52;

    cout << "Displaying the Array Contents" << endl;
    for(int i = 0; i < 3; i++)
        cout << "Name: " << StruArray[i].Name << "\tAge: " << StruArray[i].Age << endl;

    return 0;
}
```



```
Displaying the Array Contents
Name: Harvey    Age: 33
Name: Sally     Age: 26
Name: Jeff      Age: 52
```