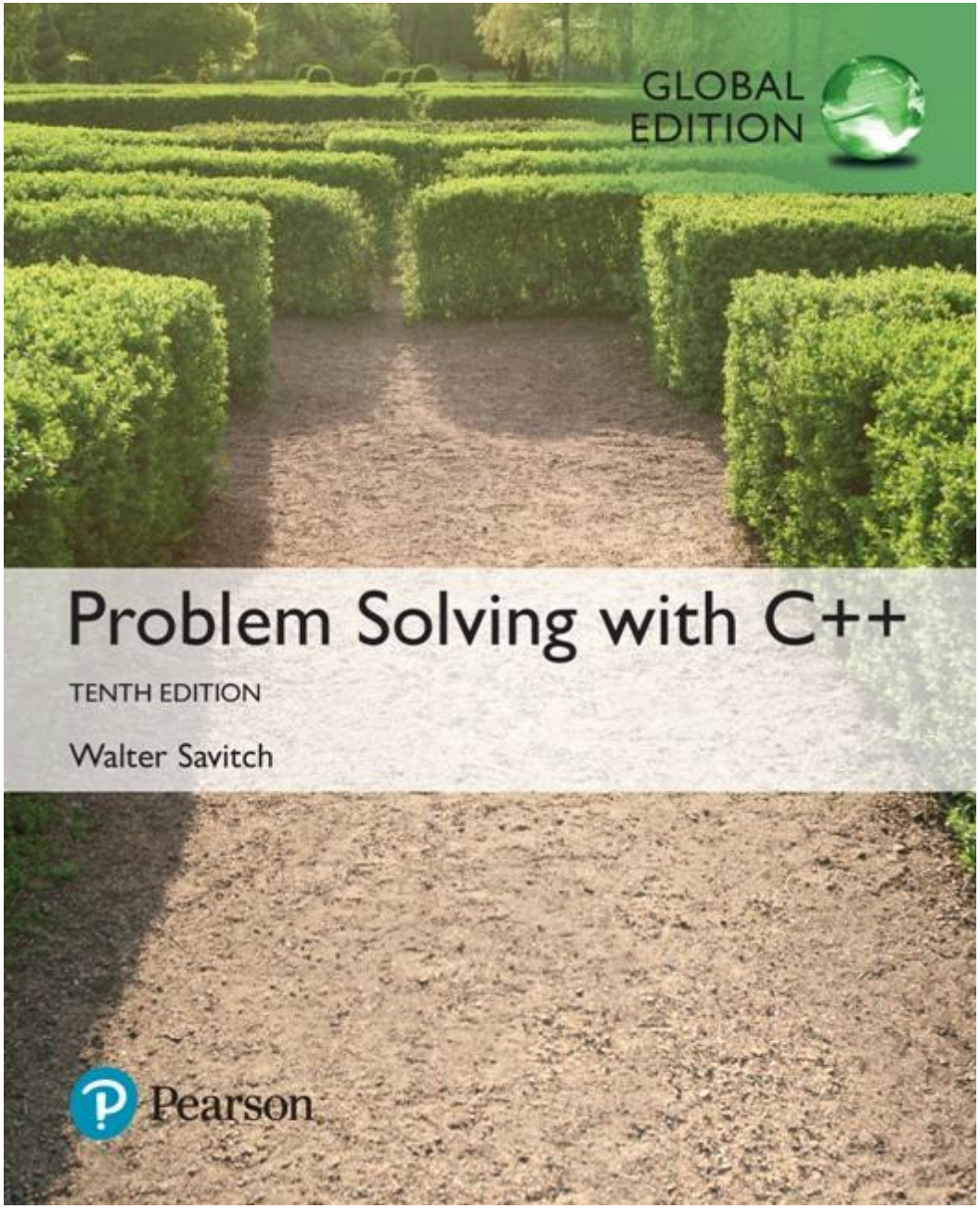
Ahram Canadian University

Fundamental of programming 1

Dr. Marian Wagdy

Lecture 5


GLOBAL
EDITION
Problem Solving with C++
TENTH EDITION
Walter Savitch
Pearson

Chapter 3

3.1 Array

3.2 References and pointer

3.1 Array

Introduction to Arrays

- An array is used to process a collection of data of the same type
  - Examples: A list of names
    A list of temperatures
- Why do we need arrays?
  - Imagine keeping track of 5 test scores, or 100, or 1000 in memory
    - How would you name all the variables?
    - How would you process each of the variables?

Declaring an Array

- An array, named score, containing five variables
  of type int can be declared as
  int score[5];
- This is like declaring 5 variables of type int:
  score[0], score[1], … , score[4]
- The value in brackets is called
  - A subscript
  - An index

The Array Variables

- The variables making up the array are referred to as
  - Indexed variables
  - Subscripted variables
  - Elements of the array
- The number of indexed variables in an array is the declared size, or size, of the array
  - The largest index is one less than the size
  - The first index value is zero

Array Variable Types

- An array can have indexed variables of any type
- All indexed variables in an array are of the same type
  - This is the base type of the array
- An indexed variable can be used anywhere an ordinary variable of the base type is used

Using [ ] With Arrays

- In an array declaration, [ ]'s enclose the size of the array such as this array of 5 integers:

  int score [5];

- When referring to one of the indexed variables, the [ ]'s enclose a number identifying one of the indexed variables
  - score[3] is one of the indexed variables
  - The value in the [ ]'s can be any expression that evaluates to one of the integers 0 to (size -1)

Indexed Variable Assignment

- To assign a value to an indexed variable, use the assignment operator:

```
int n = 2;
score[n + 1] = 99;
```

  - In this example, variable score[3] is assigned 99

Loops And Arrays

- for-loops are commonly used to step through arrays

First index is 0

Last index is (size – 1)

– Example:

```
for (i = 0; i < 5; i++)
  {
    cout << score[i] << " off by "
     << (max – score[i]) << endl;
  }
```

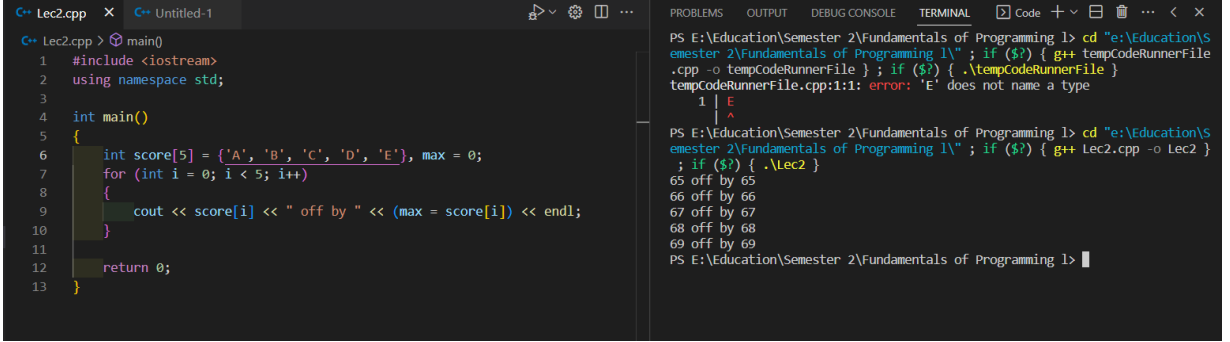
could display the difference between each score and the maximum score stored in an array

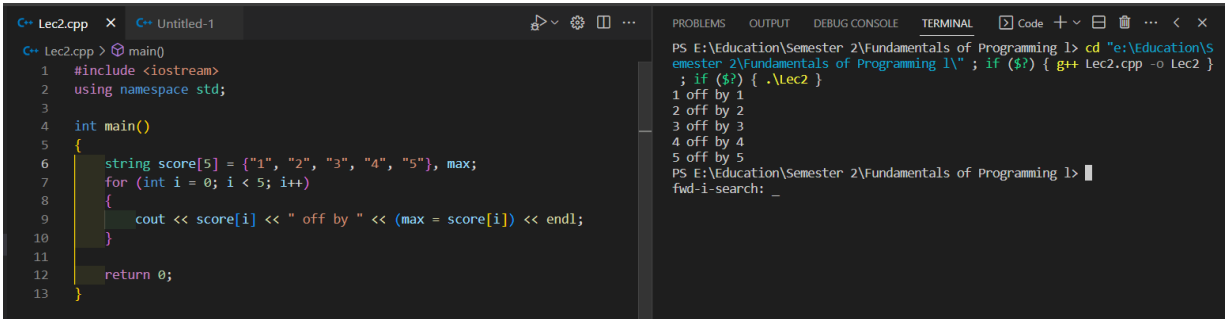
Program Using an Array

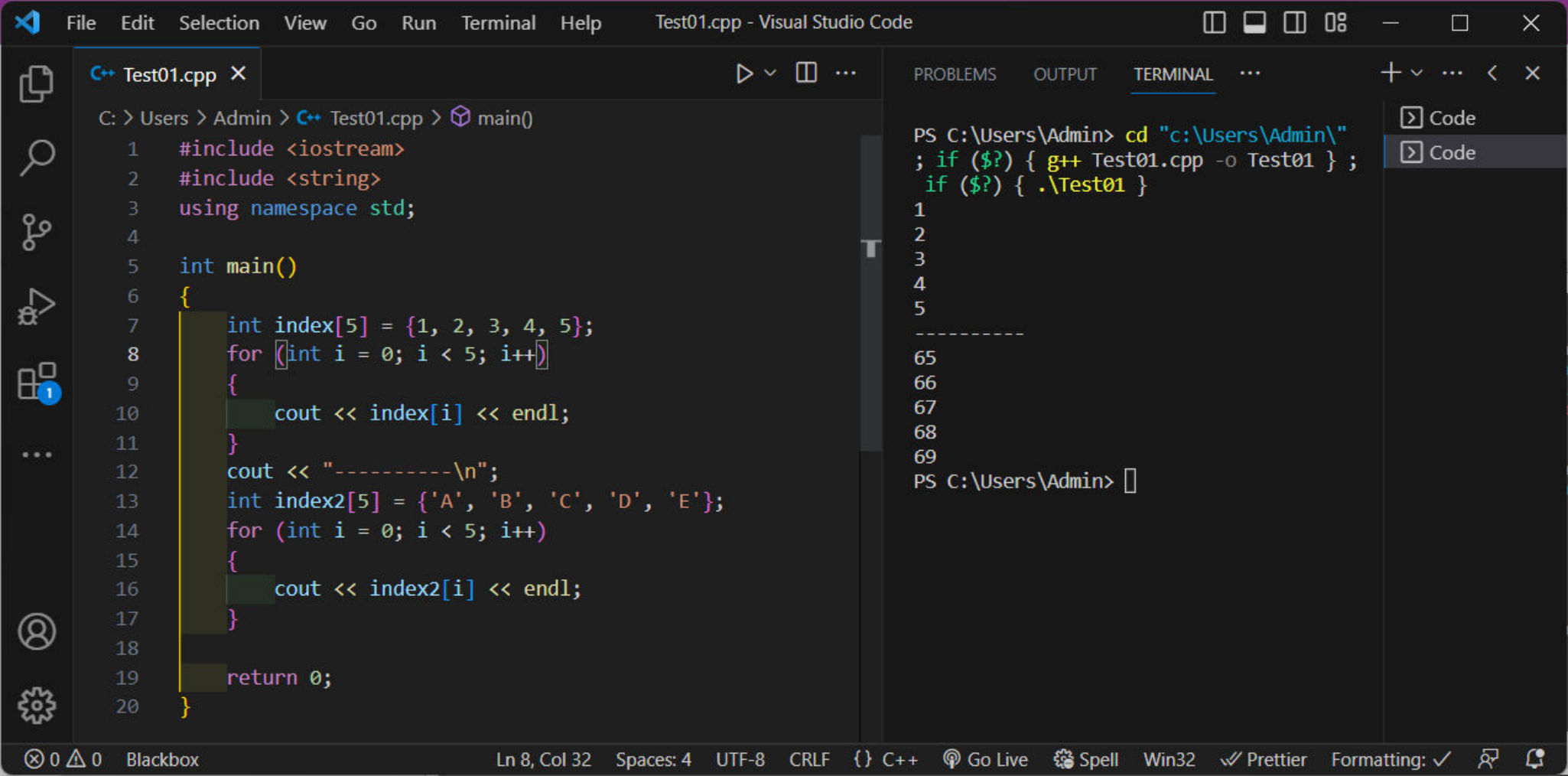
```
//Reads in 5 scores and shows how much each
//score differs from the highest score.
#include <iostream>

int main( )
{
    using namespace std;
    int i, score[5], max;

    cout << "Enter 5 scores:\n";
    cin >> score[0];
    max = score[0];
    for (i = 1; i < 5; i++)
    {
        cin >> score[i];
        if (score[i] > max)
            max = score[i];
        //max is the largest of the values score[0],..., score[i].
    }

    cout << "The highest score is " << max << endl
         << "The scores and their\n"
         << "differences from the highest are:\n";
    for (i = 0; i < 5; i++)
        cout << score[i] << " off by "
             << (max - score[i]) << endl;

    return 0;
}
```

Sample Dialogue

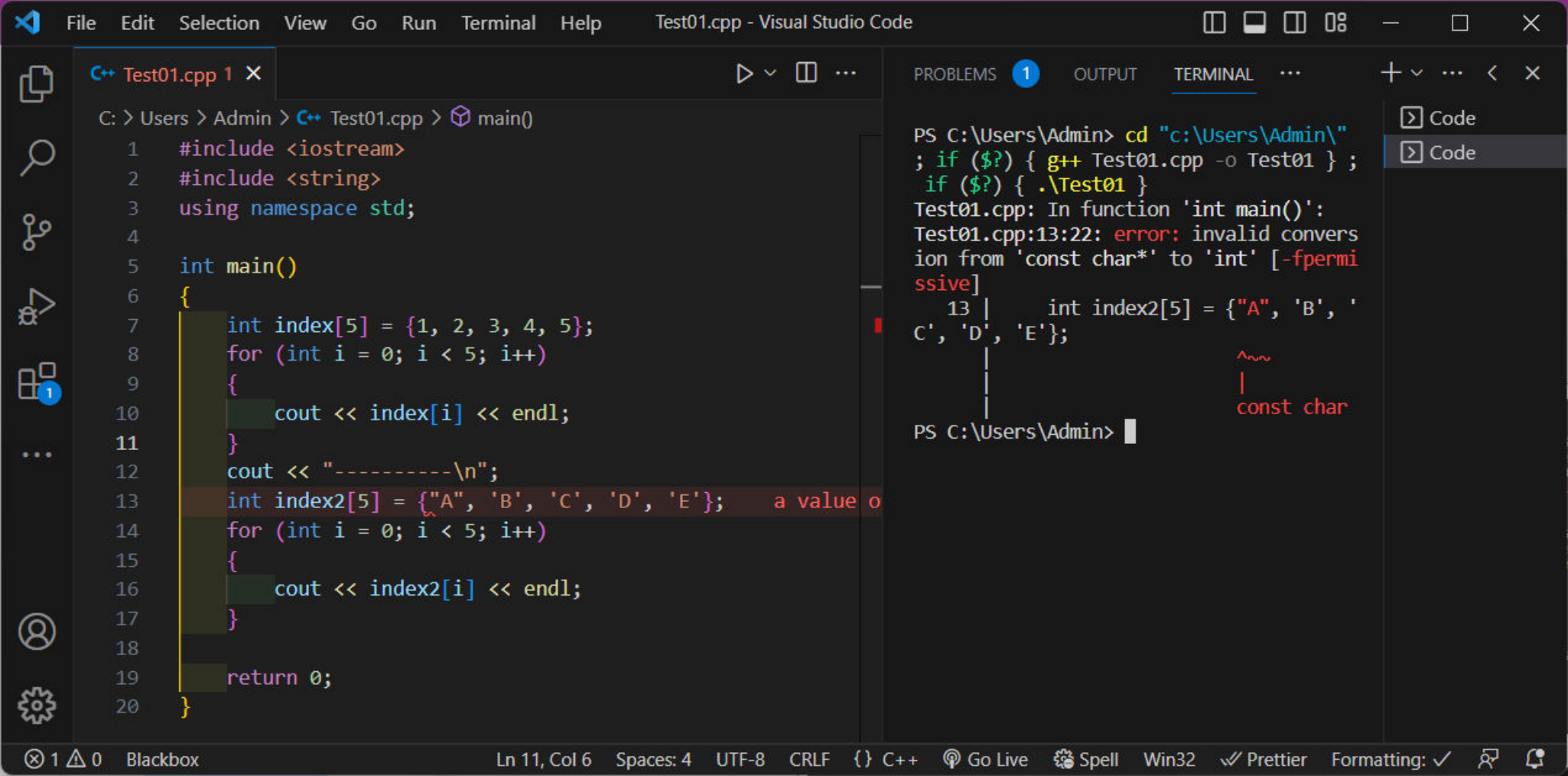
```
Enter 5 scores:
5 9 2 10 6
The highest score is 10
The scores and their
differences from the highest are:
5 off by 5
9 off by 1
2 off by 8
10 off by 0
6 off by 4
```

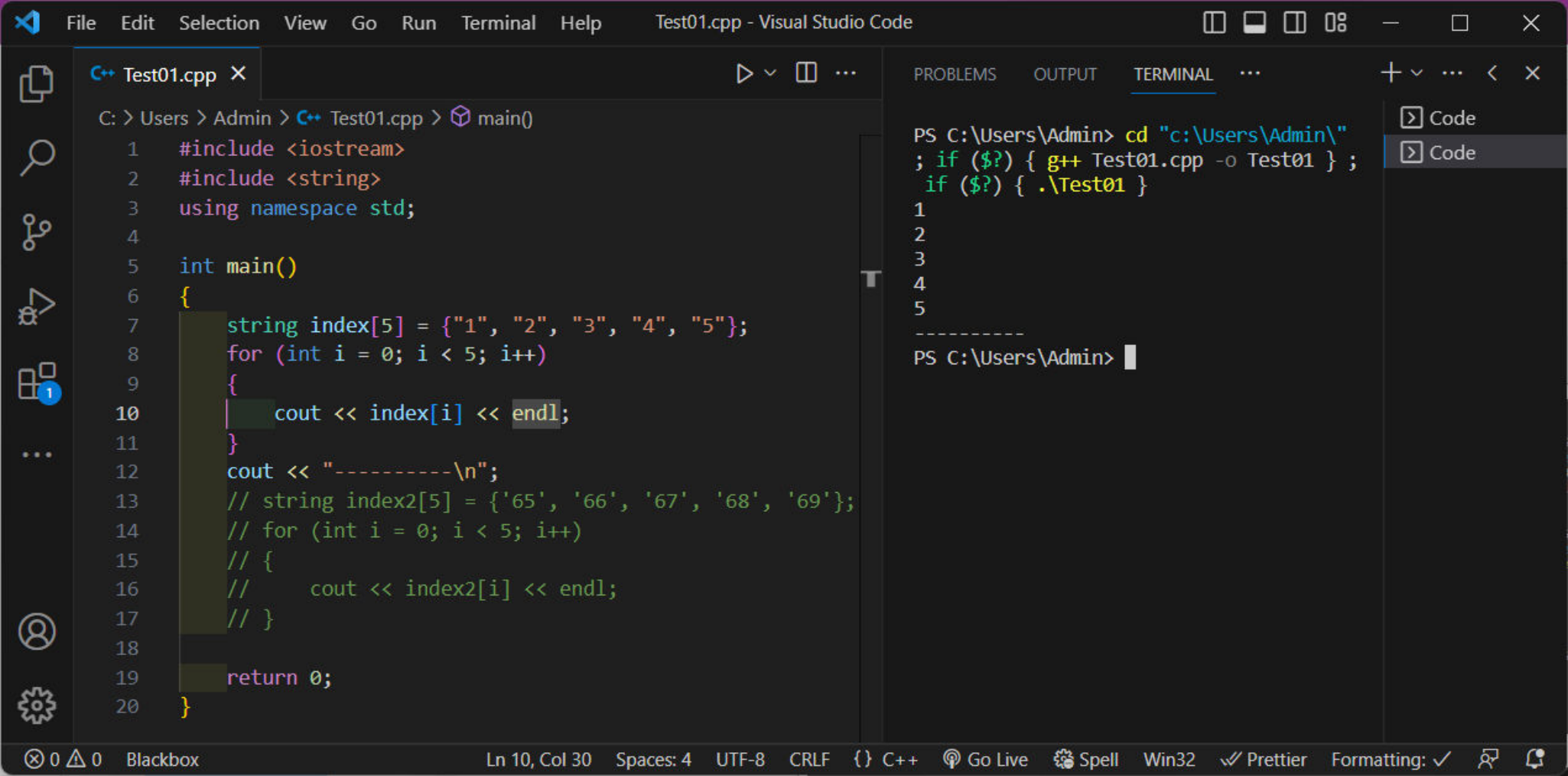

Lec2.cpp
Untitled-1
Lec2.cpp > main()
#include <iostream>
using namespace std;
int main()
{
int score[5] = {'A', 'B', 'C', 'D', 'E'}, max = 0;
for (int i = 0; i < 5; i++)
{
cout << score[i] << " off by " << (max = score[i]) << endl;
}
return 0;
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?) { g++ tempCodeRunnerFile.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
tempCodeRunnerFile.cpp:1:1: error: 'E' does not name a type
1 | E
| ^
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?) { g++ Lec2.cpp -o Lec2 } ; if ($?) { .\Lec2 }
65 off by 65
66 off by 66
67 off by 67
68 off by 68
69 off by 69
PS E:\Education\Semester 2\Fundamentals of Programming 1>


Lec2.cpp
Untitled-1
Lec2.cpp > main()
#include <iostream>
using namespace std;
int main()
{
string score[5] = {"1", "2", "3", "4", "5"}, max;
for (int i = 0; i < 5; i++)
{
cout << score[i] << " off by " << (max = score[i]) << endl;
}
return 0;
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Code
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?) { g++ Lec2.cpp -o Lec2 } ; if ($?) { .\Lec2 }
1 off by 1
2 off by 2
3 off by 3
4 off by 4
5 off by 5
PS E:\Education\Semester 2\Fundamentals of Programming 1>
fwd-i-search: _

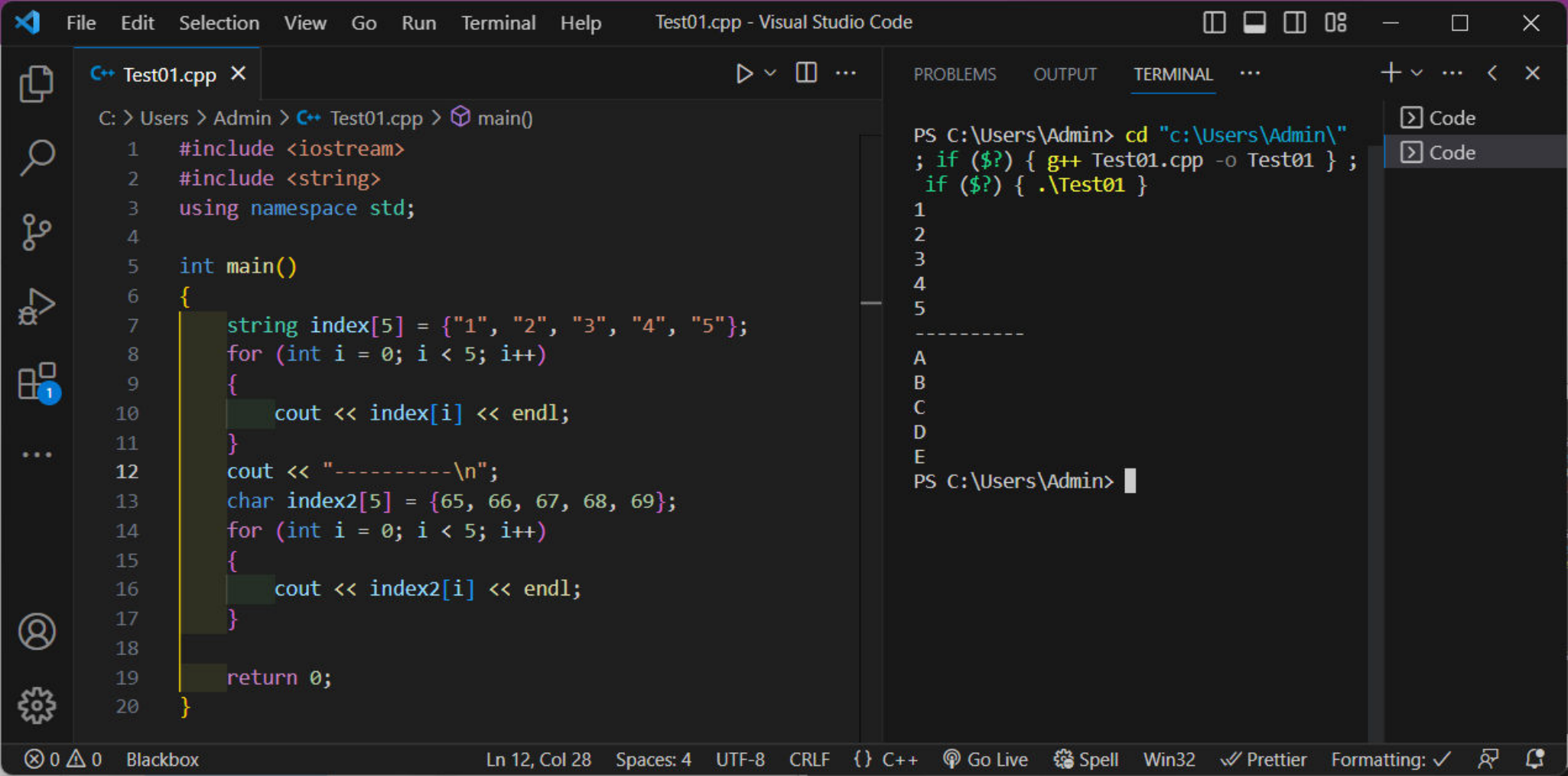

File Edit Selection View Go Run Terminal Help
Test01.cpp - Visual Studio Code
Test01.cpp
C: > Users > Admin > Test01.cpp > main()
#include <iostream>
#include <string>
using namespace std;
int main()
{
int index[5] = {1, 2, 3, 4, 5};
for (int i = 0; i < 5; i++)
{
cout << index[i] << endl;
}
cout << "----------\n";
int index2[5] = {'A', 'B', 'C', 'D', 'E'};
for (int i = 0; i < 5; i++)
{
cout << index2[i] << endl;
}
return 0;
}
PROBLEMS
OUTPUT
TERMINAL
PS C:\Users\Admin> cd "c:\Users\Admin\"
; if ($?) { g++ Test01.cpp -o Test01 } ;
if ($?) { .\Test01 }
1
2
3
4
5

65
66
67
68
69
PS C:\Users\Admin>
Code
Code
0 0 Blackbox
Ln 8, Col 32
Spaces: 4
UTF-8
CRLF
C++
Go Live
Spell
Win32
Prettier
Formatting:

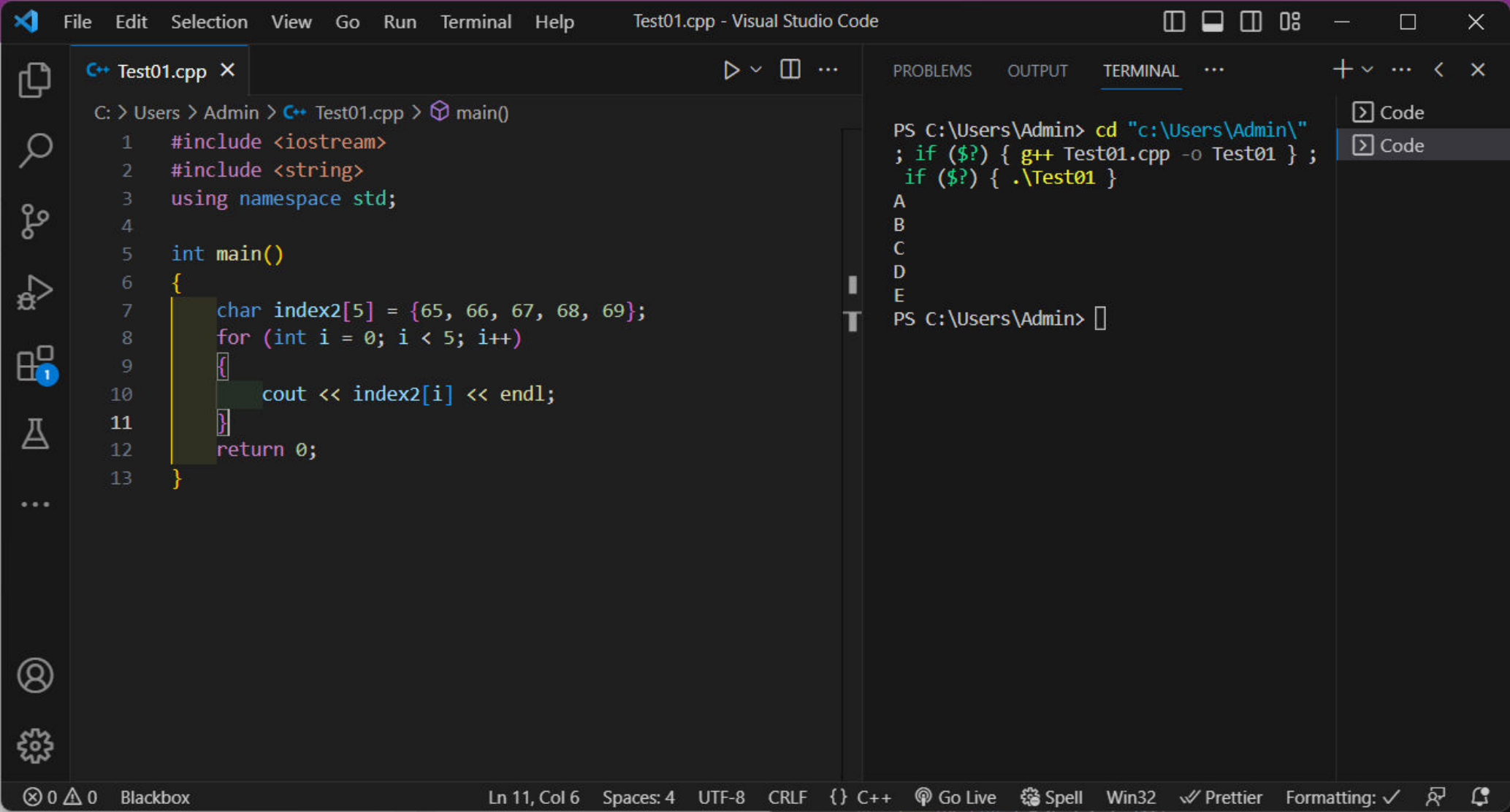

File Edit Selection View Go Run Terminal Help
Test01.cpp - Visual Studio Code
Test01.cpp 1
C: > Users > Admin > Test01.cpp > main()
#include <iostream>
#include <string>
using namespace std;
int main()
{
int index[5] = {1, 2, 3, 4, 5};
for (int i = 0; i < 5; i++)
{
cout << index[i] << endl;
}
cout << "----------\n";
int index2[5] = {"A", 'B', 'C', 'D', 'E'};
a value o
for (int i = 0; i < 5; i++)
{
cout << index2[i] << endl;
}
return 0;
}
PROBLEMS 1
OUTPUT
TERMINAL
PS C:\Users\Admin> cd "c:\Users\Admin\"
; if ($?) { g++ Test01.cpp -o Test01 } ;
if ($?) { .\Test01 }
Test01.cpp: In function 'int main()':
Test01.cpp:13:22: error: invalid conversion from 'const char*' to 'int' [-fpermissive]
13 | int index2[5] = {"A", 'B', 'C', 'D', 'E'};
| ^~~
| |
| const char
PS C:\Users\Admin>
Code
Code
1 0 Blackbox
Ln 11, Col 6
Spaces: 4
UTF-8
CRLF
C++
Go Live
Spell
Win32
Prettier
Formatting:


File Edit Selection View Go Run Terminal Help
Test01.cpp - Visual Studio Code
Test01.cpp
C: > Users > Admin > Test01.cpp > main()
#include <iostream>
#include <string>
using namespace std;
int main()
{
string index[5] = {"1", "2", "3", "4", "5"};
for (int i = 0; i < 5; i++)
{
cout << index[i] << endl;
}
cout << "----------\n";
// string index2[5] = {'65', '66', '67', '68', '69'};
// for (int i = 0; i < 5; i++)
// {
// cout << index2[i] << endl;
// }
return 0;
}
PROBLEMS
OUTPUT
TERMINAL
PS C:\Users\Admin> cd "c:\Users\Admin\"
; if ($?) { g++ Test01.cpp -o Test01 } ;
if ($?) { .\Test01 }
1
2
3
4
5

PS C:\Users\Admin>
Code
Code
0 0 Blackbox
Ln 10, Col 30
Spaces: 4
UTF-8
CRLF
C++
Go Live
Spell
Win32
Prettier
Formatting:


File Edit Selection View Go Run Terminal Help
Test01.cpp - Visual Studio Code
Test01.cpp
C: > Users > Admin > Test01.cpp > main()
#include <iostream>
#include <string>
using namespace std;
int main()
{
string index[5] = {"1", "2", "3", "4", "5"};
for (int i = 0; i < 5; i++)
{
cout << index[i] << endl;
}
cout << "----------\n";
char index2[5] = {65, 66, 67, 68, 69};
for (int i = 0; i < 5; i++)
{
cout << index2[i] << endl;
}
return 0;
}
PROBLEMS
OUTPUT
TERMINAL
PS C:\Users\Admin> cd "c:\Users\Admin\"
; if ($?) { g++ Test01.cpp -o Test01 } ;
if ($?) { .\Test01 }
1
2
3
4
5

A
B
C
D
E
PS C:\Users\Admin>
Code
Code
0 0 Blackbox
Ln 12, Col 28
Spaces: 4
UTF-8
CRLF
C++
Go Live
Spell
Win32
Prettier
Formatting:


File Edit Selection View Go Run Terminal Help
Test01.cpp - Visual Studio Code
Test01.cpp
C: > Users > Admin > Test01.cpp > main()
#include <iostream>
#include <string>
using namespace std;
int main()
{
char index2[5] = {65, 66, 67, 68, 69};
for (int i = 0; i < 5; i++)
{
cout << index2[i] << endl;
}
return 0;
}
PROBLEMS
OUTPUT
TERMINAL
PS C:\Users\Admin> cd "c:\Users\Admin\"
; if ($?) { g++ Test01.cpp -o Test01 } ;
if ($?) { .\Test01 }
A
B
C
D
E
PS C:\Users\Admin>
Code
Code
0 0 Blackbox
Ln 11, Col 6 Spaces: 4 UTF-8 CRLF {} C++ Go Live Spell Win32 Prettier Formatting:

Constants and Arrays

- Use constants to declare the size of an array
  - Using a constant allows your code to be easily altered for use on a smaller or larger set of data
    - Example:

```
const int  NUMBER_OF_STUDENTS = 50;
int score[NUMBER_OF_STUDENTS];
…
for ( i = 0; i < NUMBER_OF_STUDENTS;
i++)
cout << score[i] << " off by "
        << (max – score[i]) << endl;
```

    - Only the value of the constant must be changed to make this code work for any number of students

Lec2.cpp 1 | Untitled-1

Lec2.cpp > main()

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

int main()
{
    int x = 5;
    int score[x] = {1, 2, 3, 4, 5}, max;    expression must have a constant value
    for (int i = 0; i < 5; i++)
    {
        cout << score[i] << " off by " << (max = score[i]) << endl;
    }

    return 0;
}
```

Lec2.cpp 1 | Untitled-1

Lec2.cpp > main()

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

int main()
{
    const int look = 2;
    int score[look] = {1, 2, 3, 4, 5}, max;    too many initializer values
    for (int i = 0; i < 5; i++)
    {
        cout << score[i] << " off by " << (max = score[i]) << endl;
    }

    return 0;
}
```

Lec2.cpp | Untitled-1

Lec2.cpp > main()

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

int main()
{
    const int look = 5;
    int score[look] = {1, 2, 3, 4, 5}, max;
    for (int i = 0; i < 5; i++)
    {
        cout << score[i] << " off by " << (max = score[i]) << endl;
    }

    return 0;
}
```

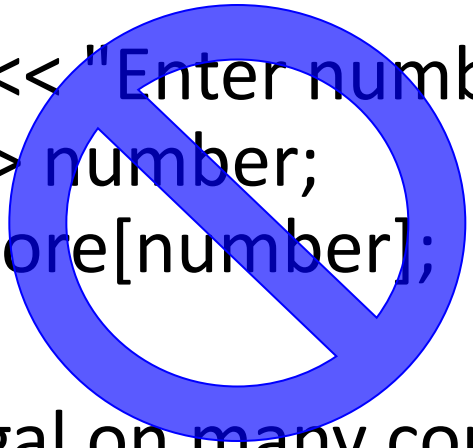
PROBLEMS | TERMINAL | Code

```
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e
:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?
) { g++ Lec2.cpp -o Lec2 } ; if ($?) { .\Lec2 }
1 off by 1
2 off by 2
3 off by 3
4 off by 4
5 off by 5
PS E:\Education\Semester 2\Fundamentals of Programming 1>
```

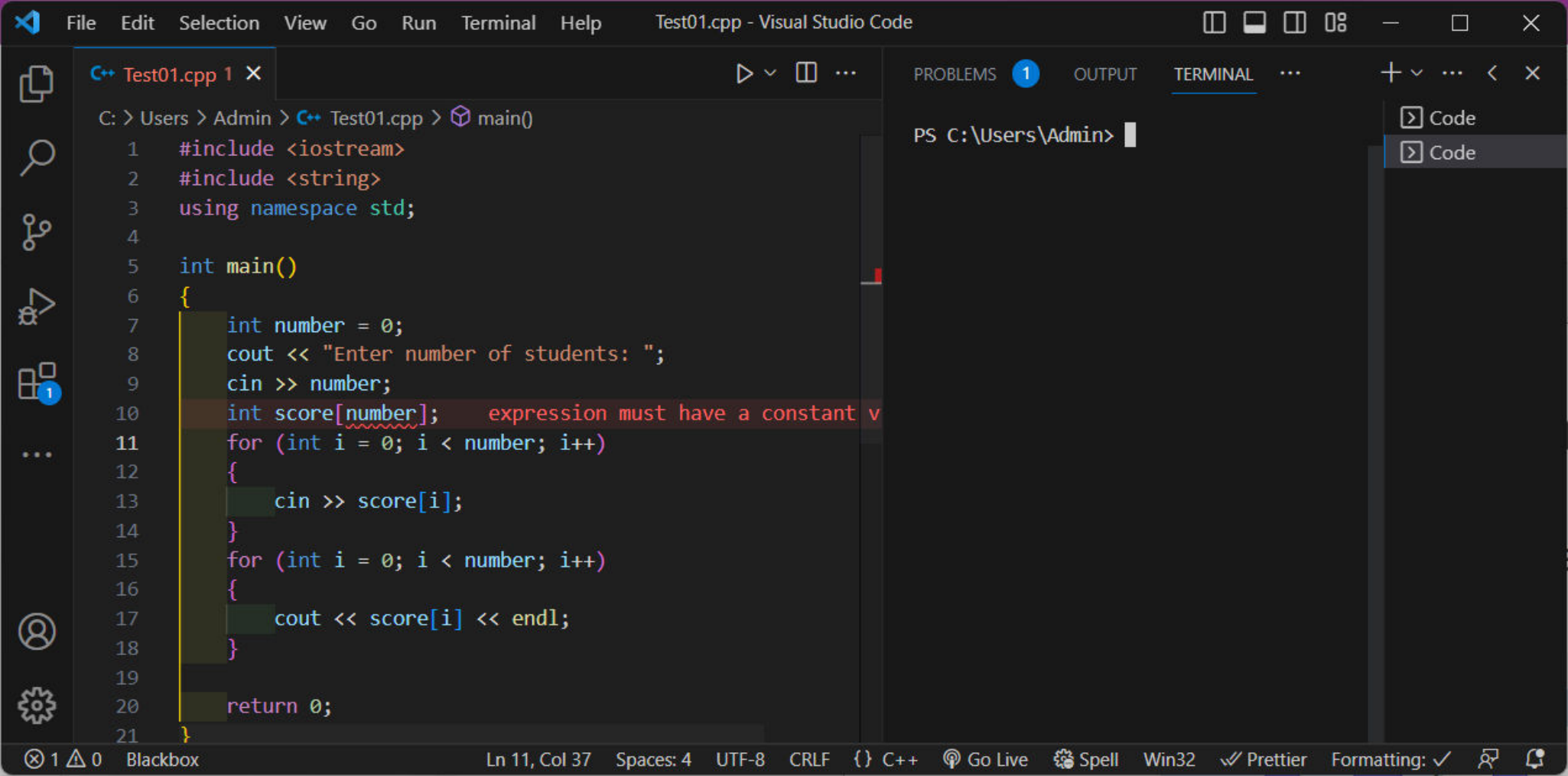
Variables and Declarations

- Some compilers do not allow the use of a variable to declare the size of an array

Example:
```
cout << "Enter number of students: ";
cin >> number;
int score[number];
```



- This code is illegal on many compilers
- Later we will see dynamic arrays which supports this idea


File Edit Selection View Go Run Terminal Help
Test01.cpp - Visual Studio Code
Test01.cpp 1
C: > Users > Admin > Test01.cpp > main()
#include <iostream>
#include <string>
using namespace std;
int main()
{
int number = 0;
cout << "Enter number of students: ";
cin >> number;
int score[number]; expression must have a constant v
for (int i = 0; i < number; i++)
{
cin >> score[i];
}
for (int i = 0; i < number; i++)
{
cout << score[i] << endl;
}
return 0;
}
PROBLEMS 1
OUTPUT
TERMINAL
PS C:\Users\Admin>
Code
Code
1 0 Blackbox
Ln 11, Col 37
Spaces: 4
UTF-8
CRLF
C++
Go Live
Spell
Win32
Prettier
Formatting:

Array Declaration Syntax

- To declare an array, use the syntax:
  Type_Name Array_Name[Declared_Size];
  - Type_Name can be any type
  - Declared_Size can be a constant to make your program more versatile
- Once declared, the array consists of the indexed variables:
  Array_Name[0] to Array_Name[Declared_Size -1]

Computer Memory

- Computer memory consists of numbered locations called bytes
  - A byte's number is its address
- A simple variable is stored in consecutive bytes
  - The number of bytes depends on the variable's type
- A variable's address is the address of its first byte

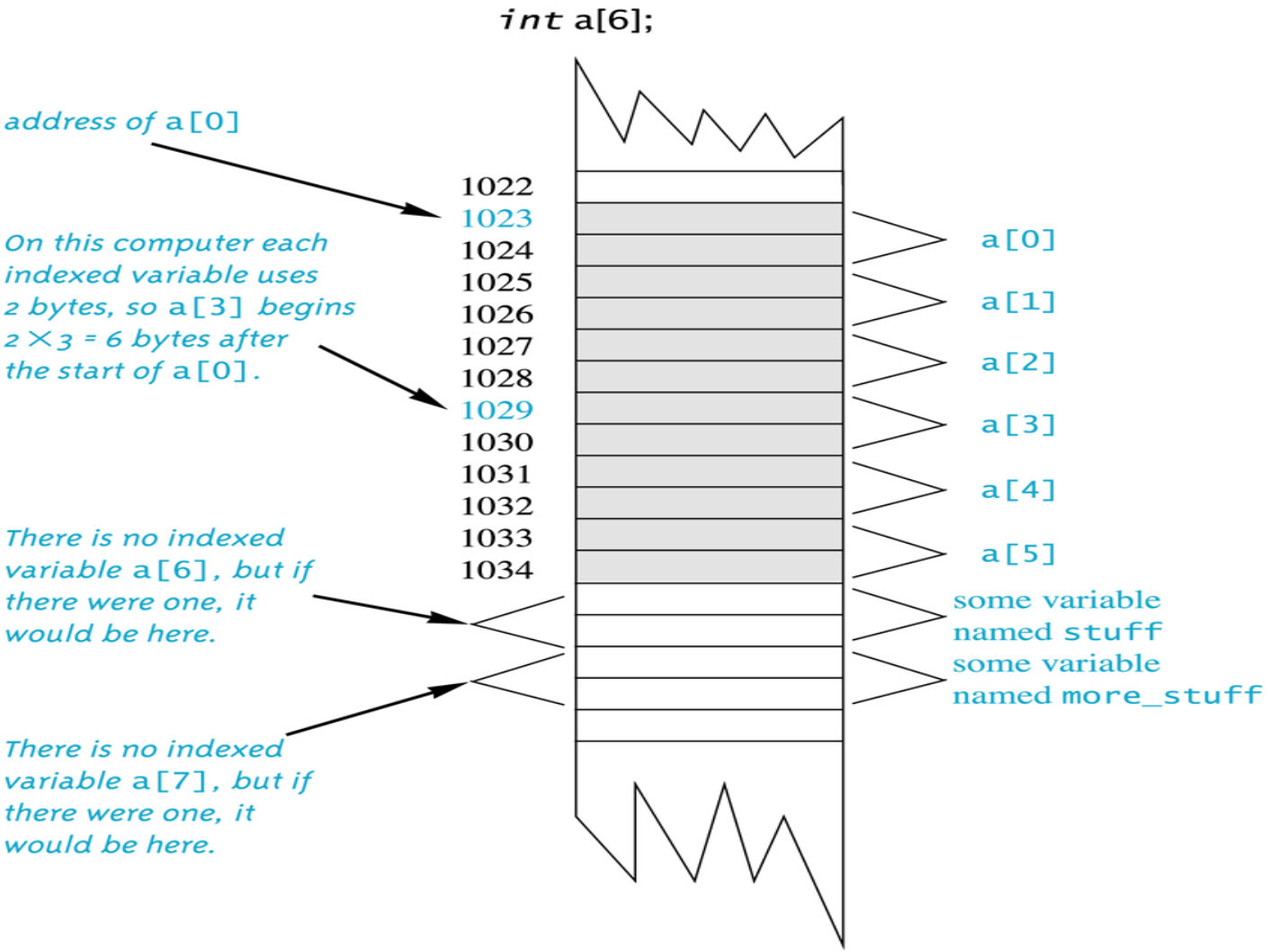
Arrays and Memory

- Declaring the array int a[6]
  - Reserves memory for six variables of type int
  - The variables are stored one after another
  - The address of a[0] is remembered
    - The addresses of the other indexed variables is not remembered
  - To determine the address of a[3]
    - Start at a[0]
    - Count past enough memory for three integers to find a[3]

Display 3.1

Display 3.1

An Array in Memory



Lec2.cpp Untitled-1

Lec2.cpp > main()

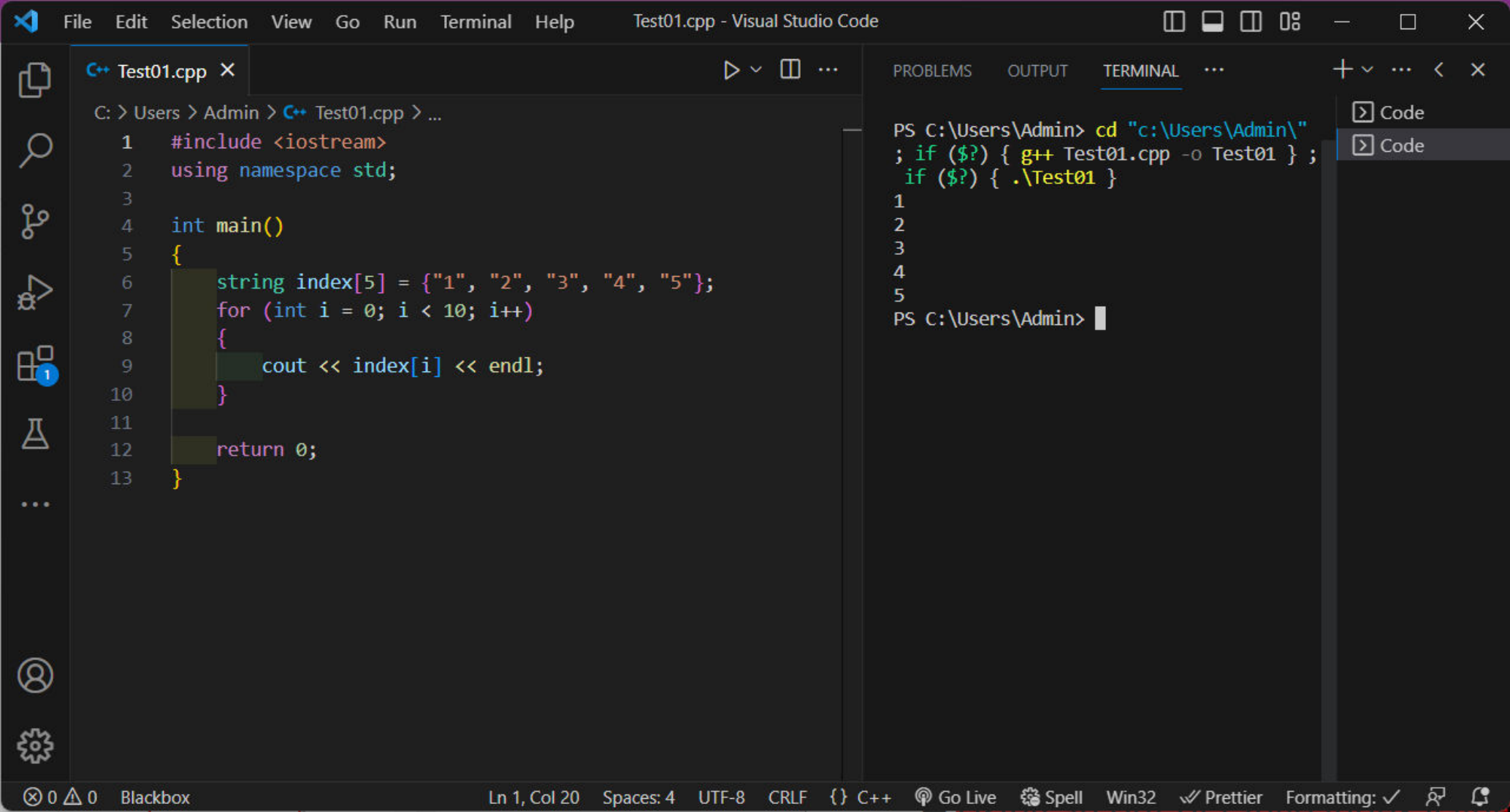
```cpp
#include <iostream>
using namespace std;

int main()
{
    const int look = 5;
    int score[look] = {1, 2, 3, 4, 5}, max;
    for (int i = 0; i < 7; i++)
    {
        cout << score[i] << " off by " << (max = score[i]) << endl;
    }

    return 0;
}
```

PROBLEMS TERMINAL ... Code

```
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e
:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?
) { g++ Lec2.cpp -o Lec2 } ; if ($?) { .\Lec2 }
1 off by 1
2 off by 2
3 off by 3
4 off by 4
5 off by 5
5 off by 5
5 off by 5
PS E:\Education\Semester 2\Fundamentals of Programming 1>
```

Array Index Out of Range

- A common error is using a nonexistent index
  - Index values for int a[6] are the values 0 through 5
  - An index value not allowed by the array declaration is out of range
  - Using an out of range index value doe not produce an error message!


File Edit Selection View Go Run Terminal Help
Test01.cpp - Visual Studio Code
Test01.cpp
C: > Users > Admin > Test01.cpp > ...
#include <iostream>
using namespace std;
int main()
{
string index[5] = {"1", "2", "3", "4", "5"};
for (int i = 0; i < 10; i++)
{
cout << index[i] << endl;
}
return 0;
}
PROBLEMS
OUTPUT
TERMINAL
PS C:\Users\Admin> cd "c:\Users\Admin\"
; if ($?) { g++ Test01.cpp -o Test01 } ;
if ($?) { .\Test01 }
1
2
3
4
5
PS C:\Users\Admin>
Code
Code
0 0 Blackbox
Ln 1, Col 20
Spaces: 4
UTF-8
CRLF
{} C++
Go Live
Spell
Win32
Prettier
Formatting:

Out of Range Problems

- If an array is declared as: int a[6];
  and an integer is declared as: int i = 7;
- Executing the statement a[i] = 238; causes…
  - The computer to calculate the address of the illegal a[7]
  - (This address could be where some other variable is stored)
  - The value 238 is stored at the address calculated for a[7]
  - No warning is given!

Initializing Arrays

- To initialize an array when it is declared
  - The values for the indexed variables are enclosed in braces and separated by commas
- Example: int children[3] = { 2, 12, 1 };
  Is equivalent to:

```
int children[3];
children[0] = 2;
children[1] = 12;
children[2] = 1;
```

Default Values

- If too few values are listed in an initialization statement
  - The listed values are used to initialize the first of the indexed variables
  - The remaining indexed variables are initialized to a zero of the base type
  - Example: int a[10] = {5, 5};
    initializes a[0] and a[1] to 5 and
    a[2] through a[9] to 0

Lec2.cpp | Untitled-1

Lec2.cpp > main()

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

int main()
{
    int score[5] = {1, 2}, max;
    for (int i = 0; i < 5; i++)
    {
        cout << score[i] << " off by " << (max = score[i]) << endl;
    }
    return 0;
}
```

PROBLEMS | TERMINAL | Code

```
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e
:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?
) { g++ Lec2.cpp -o Lec2 } ; if ($?) { .\Lec2 }
1 off by 1
2 off by 2
0 off by 0
0 off by 0
0 off by 0
584 off by 584
584 off by 584
PS E:\Education\Semester 2\Fundamentals of Programming 1>
```

Un-initialized Arrays

- If no values are listed in the array declaration, some compilers will initialize each variable to a
  zero of the base type
  - DO NOT DEPEND ON THIS!

Lec2.cpp | Untitled-1

Lec2.cpp > main()

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

int main()
{
    int score[] = {1, 2, 3, 4}, max;
    for (int i = 0; i < 5; i++)
    {
        cout << score[i] << " off by " << (max = score[i]) << endl;
    }
    return 0;
}
```

PROBLEMS TERMINAL ... Code

```
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e
:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?
) { g++ Lec2.cpp -o Lec2 } ; if ($?) { .\Lec2 }
1 off by 1
2 off by 2
3 off by 3
4 off by 4
124655352 off by 124655352
PS E:\Education\Semester 2\Fundamentals of Programming 1>
fwd-i-search: _
```

Range-Based For Loops

- C++11 includes a new type of for loop, the range-based for loop, that simplifies iteration over every element in an array. The syntax is shown below:

```
for (datatype varname : array)
{
        // varname is successively set to each
        // element in the array
}
```

Range-Based For Loop Example

- The following code outputs 2 4 6 8

```
int arr[ ] = {2, 4, 6, 8};
for (int x : arr)
        cout << x;
cout << endl;
```

Section 7.1 Conclusion

- Can you
  - Describe the difference between a[4] and int a[5]?
  - Show the output of

```
char symbol[3] = {'a', 'b', 'c'};
for (int index = 0; index < 3; index++)
   cout << symbol[index];
```

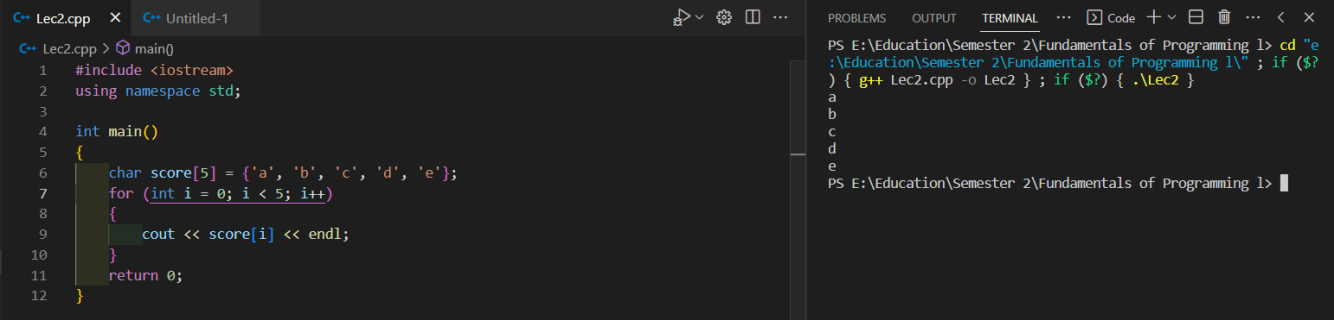
Lec2.cpp | Untitled-1

Lec2.cpp > main()

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

int main()
{
    char score[5] = {1, 2, 3, 4, 5};
    for (int i = 0; i < 5; i++)
    {
        cout << score[i] << endl;
    }
    return 0;
}
```

PROBLEMS OUTPUT TERMINAL ... Code

```
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e
:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?
) { g++ Lec2.cpp -o Lec2 } ; if ($?) { .\Lec2 }
☺
☻
♥
♦
♣
PS E:\Education\Semester 2\Fundamentals of Programming 1>
```


Lec2.cpp
Untitled-1
Lec2.cpp > main()
#include <iostream>
using namespace std;
int main()
{
char score[5] = {'a', 'b', 'c', 'd', 'e'};
for (int i = 0; i < 5; i++)
{
cout << score[i] << endl;
}
return 0;
}
PROBLEMS
OUTPUT
TERMINAL
Code
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e
:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?
) { g++ Lec2.cpp -o Lec2 } ; if ($?) { .\Lec2 }
a
b
c
d
e
PS E:\Education\Semester 2\Fundamentals of Programming 1>

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

int main() {
  int myNumbers[5] = {10, 20, 30, 40, 50};
  cout << sizeof(myNumbers);
  return 0;
}
```

```
20
```

Why did the result show 20 instead of 5, when the array contains 5 elements?

It is because the sizeof() operator returns the size of a type in bytes.

(4 bytes x 5 elements) = 20 bytes.

practice.cpp > main()

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

int main()
{
    int myNumbers[5] = {10, 20, 30, 40, 50};
    int getArrayLength = sizeof(myNumbers) / sizeof(int);
    cout << getArrayLength;

    return 0;
}
```

```
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e
:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?
) { g++ practice.cpp -o practice } ; if ($?) { .\practice }
5
PS E:\Education\Semester 2\Fundamentals of Programming 1>
```

practice.cpp

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

int main()
{
    string letters[2][4] = {
        {"A", "B", "C", "D"},
        {"E", "F", "G", "H"}};

    for (int i = 0; i < 2; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            cout << letters[i][j] << "\n";
        }
    }

    return 0;
}
```

```
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e
:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?
) { g++ practice.cpp -o practice } ; if ($?) { .\practice }
A
B
C
D
E
F
G
H
PS E:\Education\Semester 2\Fundamentals of Programming 1>
```

3.2 References and pointer

& is ampersand

* is asterisk

Lec2.cpp | Untitled-1

Lec2.cpp > main()

```cpp

int main()
{
    int x = 20;
    int *prtx = &x;
    cout << &x << endl;
    cout << prtx << endl;
    return 0;
}
```

PROBLEMS OUTPUT TERMINAL ... Code

```
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e
:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?
) { g++ Lec2.cpp -o Lec2 } ; if ($?) { .\Lec2 }
0x7e515ffae4
0x7e515ffae4
PS E:\Education\Semester 2\Fundamentals of Programming 1>
```

Lec2.cpp | Untitled-1

Lec2.cpp > main()

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

int main()
{
    int x = 20;
    int *prtx = &x;
    cout << &x << endl;
    cout << *prtx << endl;
    return 0;
}
```

PROBLEMS OUTPUT TERMINAL ... Code

```
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e
:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?
) { g++ Lec2.cpp -o Lec2 } ; if ($?) { .\Lec2 }
0x5a789ffb34
20
PS E:\Education\Semester 2\Fundamentals of Programming 1>
```

Lec2.cpp | Untitled-1

Lec2.cpp > main()

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

int main()
{
    int x = 20;
    int *prtx = &x;
    cout << &x << endl;
    cout << &prtx << endl;
    return 0;
}
```

PROBLEMS OUTPUT TERMINAL ... Code

```
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e
:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?
) { g++ Lec2.cpp -o Lec2 } ; if ($?) { .\Lec2 }
0xab569ff90c
0xab569ff900
PS E:\Education\Semester 2\Fundamentals of Programming 1>
```

Lec2.cpp Untitled-1

Lec2.cpp > main()

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

int main()
{
    int x = 20;
    int *prtx = &x;
    cout << &x << endl;
    cout << *prtx << endl;
    *prtx = 25;
    cout << *prtx << endl;
    cout << x << endl;
    return 0;
}
```

PROBLEMS OUTPUT TERMINAL ... Code

```
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e
:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?
) { g++ Lec2.cpp -o Lec2 } ; if ($?) { .\Lec2 }
0xf77d5ffdb4
20
25
25
PS E:\Education\Semester 2\Fundamentals of Programming 1>
```

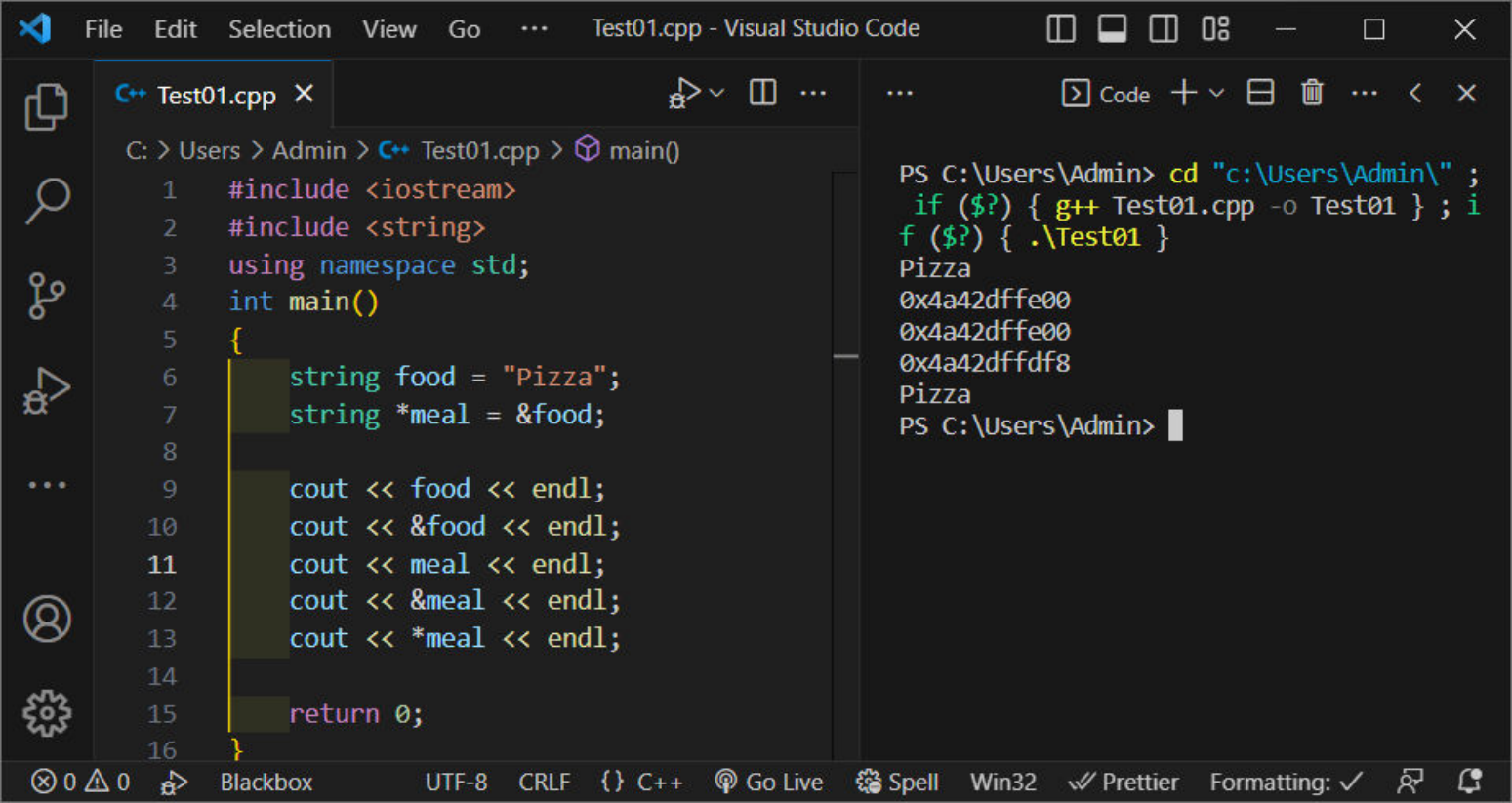
Lec2.cpp | Untitled-1

Lec2.cpp > main()

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

int main()
{
    int *prtx = new int(20);
    cout << *prtx << endl;
    return 0;
}
```

PROBLEMS OUTPUT TERMINAL ... Code

```
PS E:\Education\Semester 2\Fundamentals of Programming 1> cd "e
:\Education\Semester 2\Fundamentals of Programming 1\" ; if ($?
) { g++ Lec2.cpp -o Lec2 } ; if ($?) { .\Lec2 }
20
PS E:\Education\Semester 2\Fundamentals of Programming 1>
```


File Edit Selection View Go ··· Test01.cpp - Visual Studio Code
Test01.cpp
C: > Users > Admin > Test01.cpp > main()
#include <iostream>
#include <string>
using namespace std;
int main()
{
string food = "Pizza";
string *meal = &food;
cout << food << endl;
cout << &food << endl;
cout << meal << endl;
cout << &meal << endl;
cout << *meal << endl;
return 0;
}
Code
PS C:\Users\Admin> cd "c:\Users\Admin\" ; if ($?) { g++ Test01.cpp -o Test01 } ; if ($?) { .\Test01 }
Pizza
0x4a42dffe00
0x4a42dffe00
0x4a42dffdf8
Pizza
PS C:\Users\Admin>
0 0 Blackbox UTF-8 CRLF {} C++ Go Live Spell Win32 Prettier Formatting:

C++ References

- Creating References

  A reference variable is a "reference" to an existing variable, and it is created with the & operator:

```
string food = "Pizza";  // food variable
string &meal = food;    // reference to food
```

- Now, we can use either the variable name food or the reference name meal to refer to the food variable:

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

int main() {
  string food = "Pizza";
  string &meal = food;

  cout << food << "\n";
  cout << meal << "\n";
  return 0;
}
```

Practice #01

File Edit Selection View Go ··· Test01.cpp - Visual Studio Code

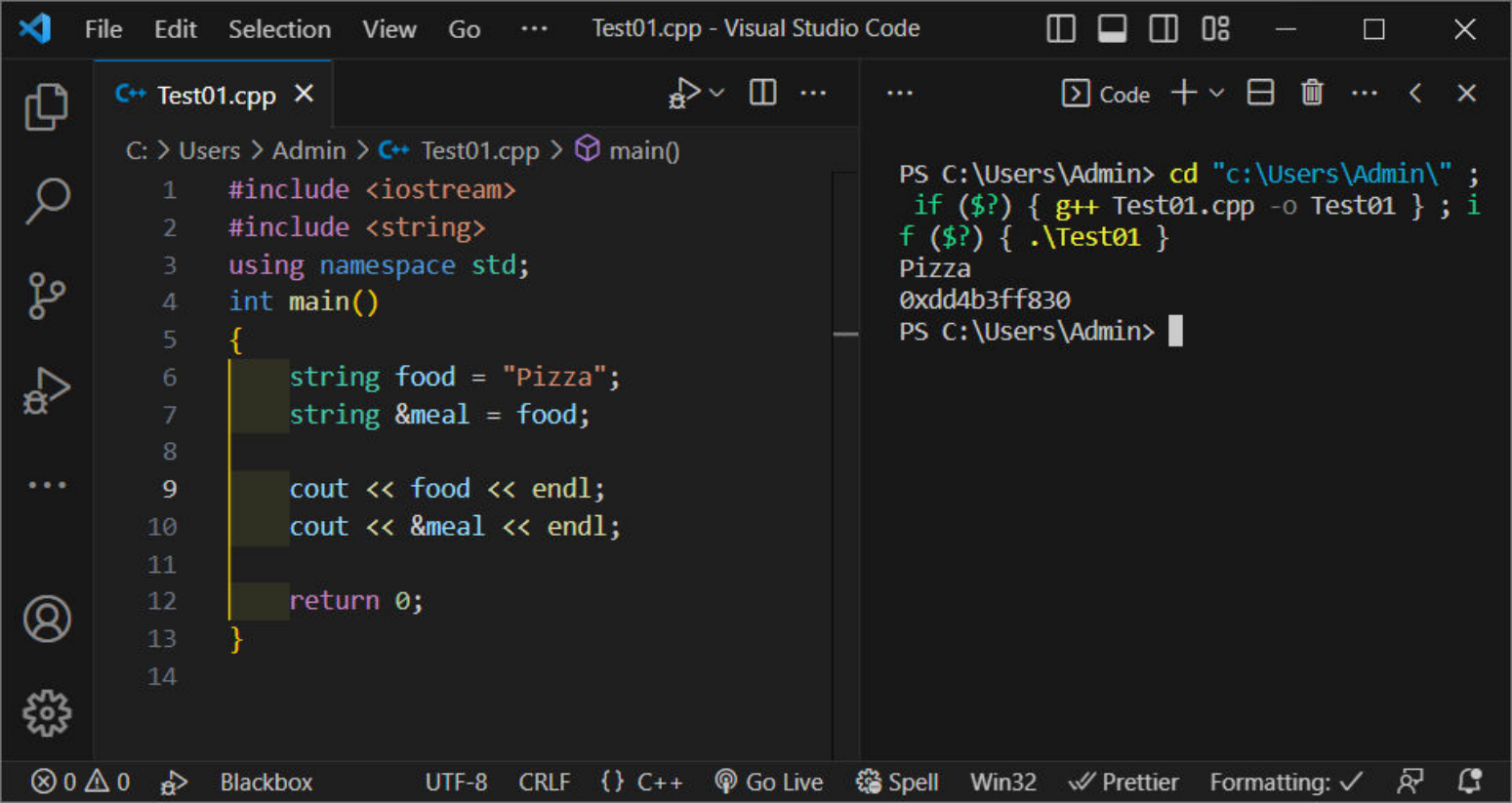
Test01.cpp

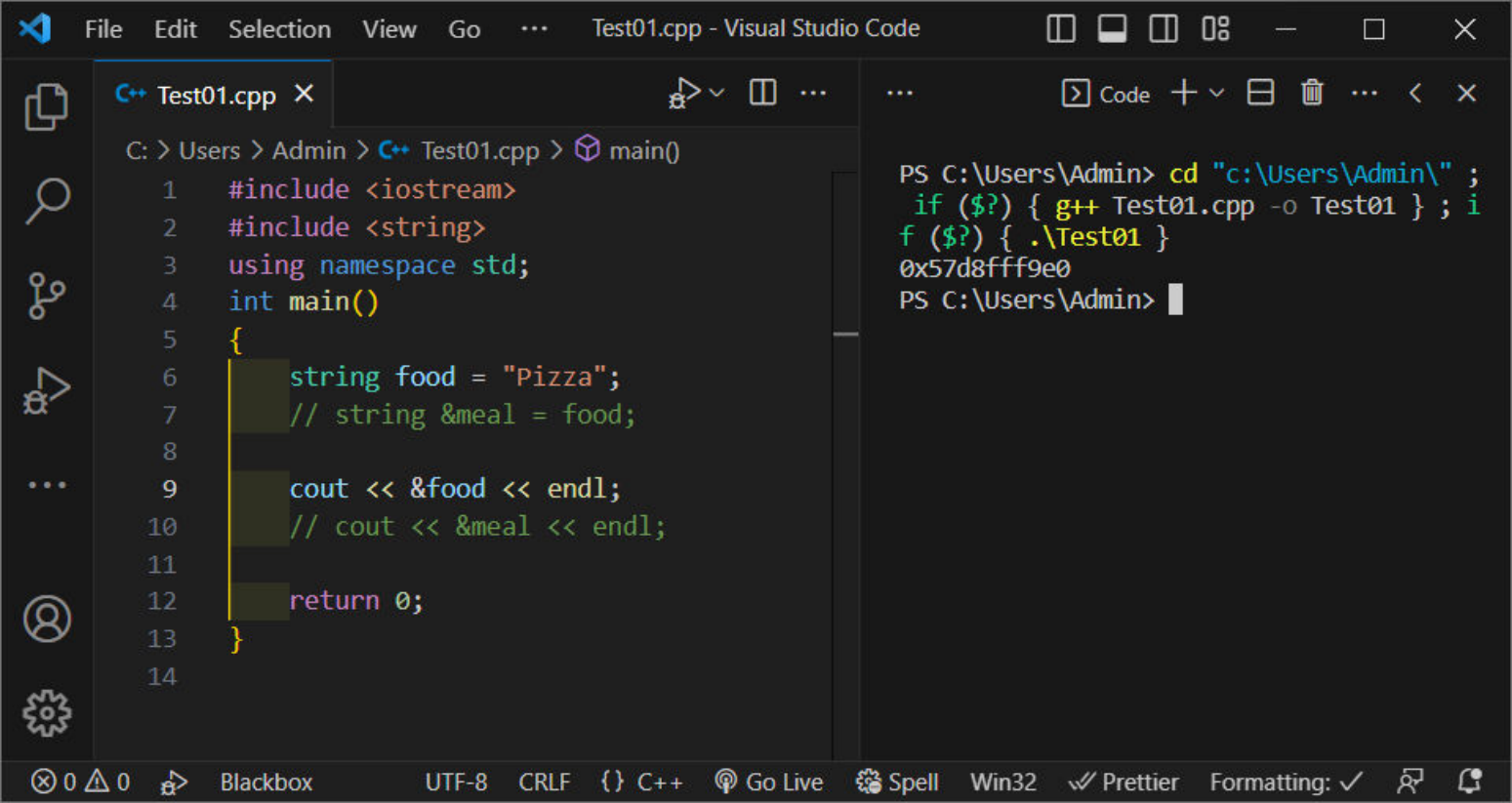
C: > Users > Admin > Test01.cpp > main()

```cpp
#include <iostream>
#include <string>
using namespace std;
int main()
{
    string food = "Pizza";
    string &meal = food;

    cout << food << endl;
    cout << meal << endl;

    return 0;
}
```

```
PS C:\Users\Admin> cd "c:\Users\Admin\" ;
 if ($?) { g++ Test01.cpp -o Test01 } ; i
f ($?) { .\Test01 }
Pizza
Pizza
PS C:\Users\Admin>
```

0 0 Blackbox UTF-8 CRLF {} C++ Go Live Spell Win32 Prettier Formatting: ✓


File Edit Selection View Go ··· Test01.cpp - Visual Studio Code
Test01.cpp
C: > Users > Admin > Test01.cpp > main()
#include <iostream>
#include <string>
using namespace std;
int main()
{
string food = "Pizza";
string &meal = food;
cout << food << endl;
cout << &meal << endl;
return 0;
}
Code
PS C:\Users\Admin> cd "c:\Users\Admin\" ; if ($?) { g++ Test01.cpp -o Test01 } ; if ($?) { .\Test01 }
Pizza
0xdd4b3ff830
PS C:\Users\Admin>
0 0 Blackbox UTF-8 CRLF {} C++ Go Live Spell Win32 Prettier Formatting:


File Edit Selection View Go ··· Test01.cpp - Visual Studio Code
Test01.cpp
C: > Users > Admin > Test01.cpp > main()
#include <iostream>
#include <string>
using namespace std;
int main()
{
string food = "Pizza";
// string &meal = food;
cout << &food << endl;
// cout << &meal << endl;
return 0;
}
Code
PS C:\Users\Admin> cd "c:\Users\Admin\" ; if ($?) { g++ Test01.cpp -o Test01 } ; if ($?) { .\Test01 }
0x57d8fff9e0
PS C:\Users\Admin>
0 0 Blackbox UTF-8 CRLF {} C++ Go Live Spell Win32 Prettier Formatting:

Memory Address

- In the example above, the & operator was used to create a reference variable. But it can also be used to get the memory address of a variable, which is the location of where the variable is stored on the computer.

- When a variable is created in C++, a memory address is assigned to the variable. And when we assign a value to the variable, it is stored in this memory address.

- To access it, use the & operator, and the result will represent where the variable is stored:

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

int main() {
  string food = "Pizza";

  cout << &food;
  return 0;
}
```

Pointers

- A pointer is a variable that stores the address of another variable, stored at an "address in computer memory"

- But *pointers* are extremely powerful for low-level (memory access) programming
- With increased power comes increased responsibility
  - Pointers allow new and more ugly types of bugs that can crash in random ways which makes them more difficult to debug
- Even with their problems, pointers are powerful programming construct

Why Have Pointers?

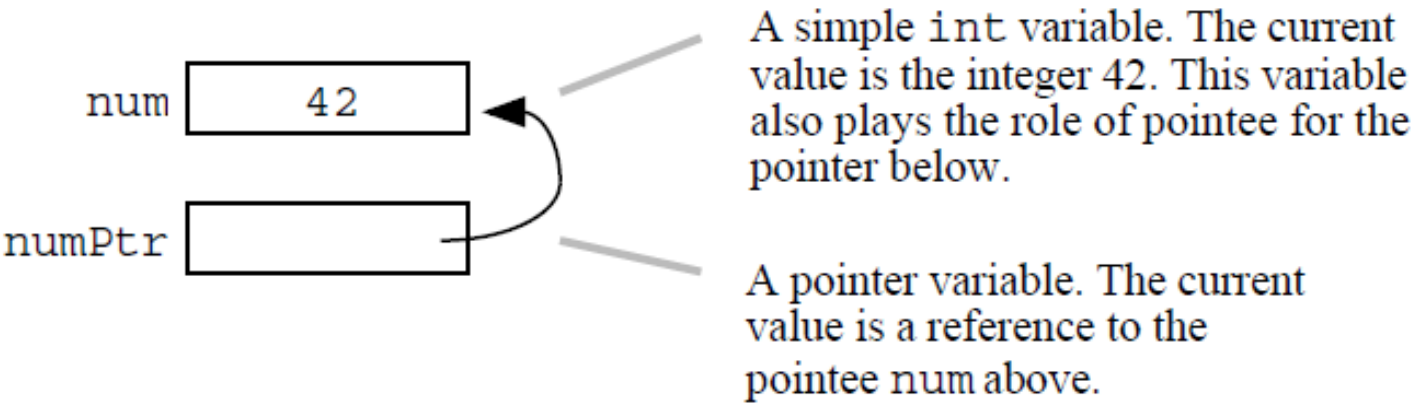
Pointers solve the following common purposes:

1. Pointers allow functions in code to share data easily. (pass by address)
2. Pointers enable complex "linked" data structures like linked lists and binary trees.
3. Pointers facilitate dynamic memory allocation

Pointer Variable

- A pointer stores a **reference** (**address**) to another value.

- The variable the pointer refers to is sometimes known as its "pointee". A pointer is a box which contains the beginning of an arrow which leads to its pointee.



Pointers in C

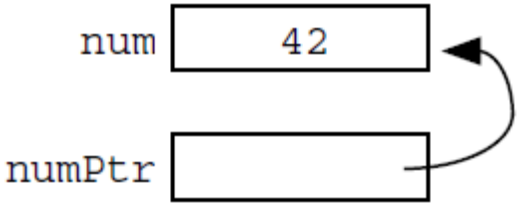
- For any data type in C, there is a corresponding pointer of same type.

- Two important operators used with pointers are;
  - Reference/Address Operator: &
  - Dereferencing Operator: *

The & Operator — Reference To

- The ***&*** operator is placed at the left of a variable, and it computes a reference address to that variable

```
int num;
int* numPtr;
num = 42;
numPtr = &num;
// Compute a reference to "num", and store it in numPtr
```



The * Operator — Dereference

- The star operator (*) dereferences a pointer.
- The * is a ***unary*** operator which goes to the left of the pointer it dereferences.
- The pointer must have a pointee, or it's a runtime error.

Pointer Variable

- Pointer variables are declared just like any other variable.
- The declaration gives the type and name of the new variable and reserves memory to hold its value.
- The declaration does not assign a pointee for the pointer — the pointer starts out with unknown value.
- The null pointer points to no location

```
int* numPtr; //Declares the pointer variable "numPtr".
// This allocates space for the pointer, but not the pointee.
```

C++ Pointers

- Creating Pointers

Example

```
string food = "Pizza"; // A food variable of type string


cout << food;  // Outputs the value of food (Pizza)
cout << &food; // Outputs the memory address of food (0x6dfed4)
```

- A **pointer** however, is a variable that **stores the memory address as its value**.
- A pointer variable points to a data type (like int or string) of the same type, and is created with the * operator.
- The address of the variable you're working with is assigned to the pointer:

Example

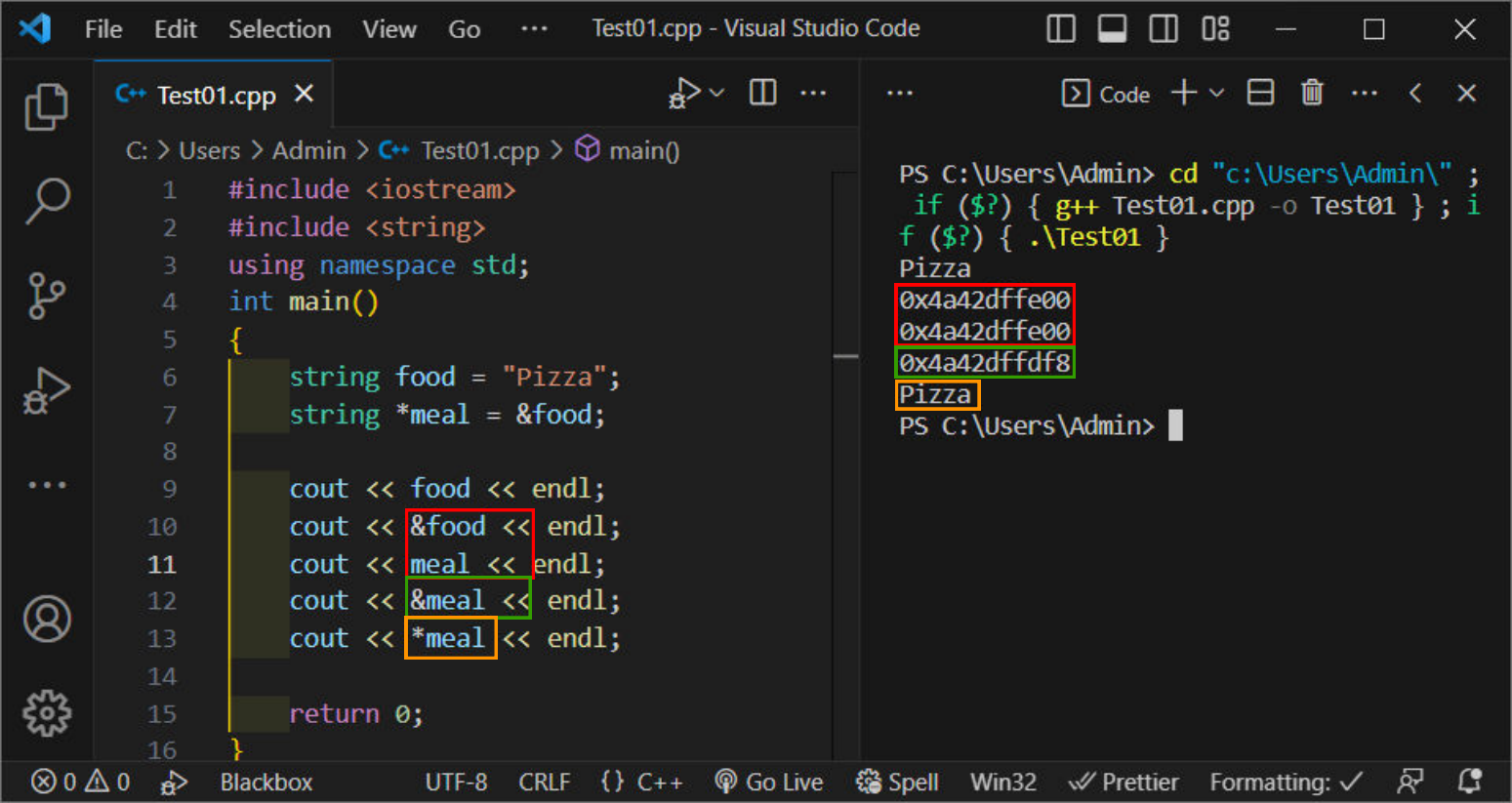
```
string food = "Pizza";  // A food variable of type string
string* ptr = &food;    // A pointer variable, with the name ptr, that stores the address of food

// Output the value of food (Pizza)
cout << food << "\n";

// Output the memory address of food (0x6dfed4)
cout << &food << "\n";

// Output the memory address of food with the pointer (0x6dfed4)
cout << ptr << "\n";
```

Get Memory Address and Value

- you can also use the pointer to get the value of the variable, by using the * operator (the **dereference** operator):

Example

```
string food = "Pizza";  // Variable declaration
string* ptr = &food;    // Pointer declaration

// Reference: Output the memory address of food with the pointer (0x6dfed4)
cout << ptr << "\n";

// Dereference: Output the value of food with the pointer (Pizza)
cout << *ptr << "\n";
```


File Edit Selection View Go ··· Test01.cpp - Visual Studio Code
Test01.cpp
C: > Users > Admin > Test01.cpp > main()
1 #include <iostream>
2 #include <string>
3 using namespace std;
4 int main()
5 {
6 string food = "Pizza";
7 string *meal = &food;
8
9 cout << food << endl;
10 cout << &food << endl;
11 cout << meal << endl;
12 cout << &meal << endl;
13 cout << *meal << endl;
14
15 return 0;
16 }
Code
PS C:\Users\Admin> cd "c:\Users\Admin\" ; if ($?) { g++ Test01.cpp -o Test01 } ; if ($?) { .\Test01 }
Pizza
0x4a42dffe00
0x4a42dffe00
0x4a42dffdf8
Pizza
PS C:\Users\Admin>
0 0 Blackbox UTF-8 CRLF {} C++ Go Live Spell Win32 Prettier Formatting: