Arrays

Ahsan Ayub

Ph.D. Student, Department of Computer Science

Graduate Research Assistant, CEROC

CSC 1300: Introduction to Programming

Tuesday, October 19, 2021

Simple Problem-Solving Tasks

Write a C++ program that will print the average of 3 scores.

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

int main()
{
        int score1 = 72;
        int score2 = 73;
        int score3 = 33;
        cout << (score1+score2+score3)/3 << endl;
        return 0;
}
```

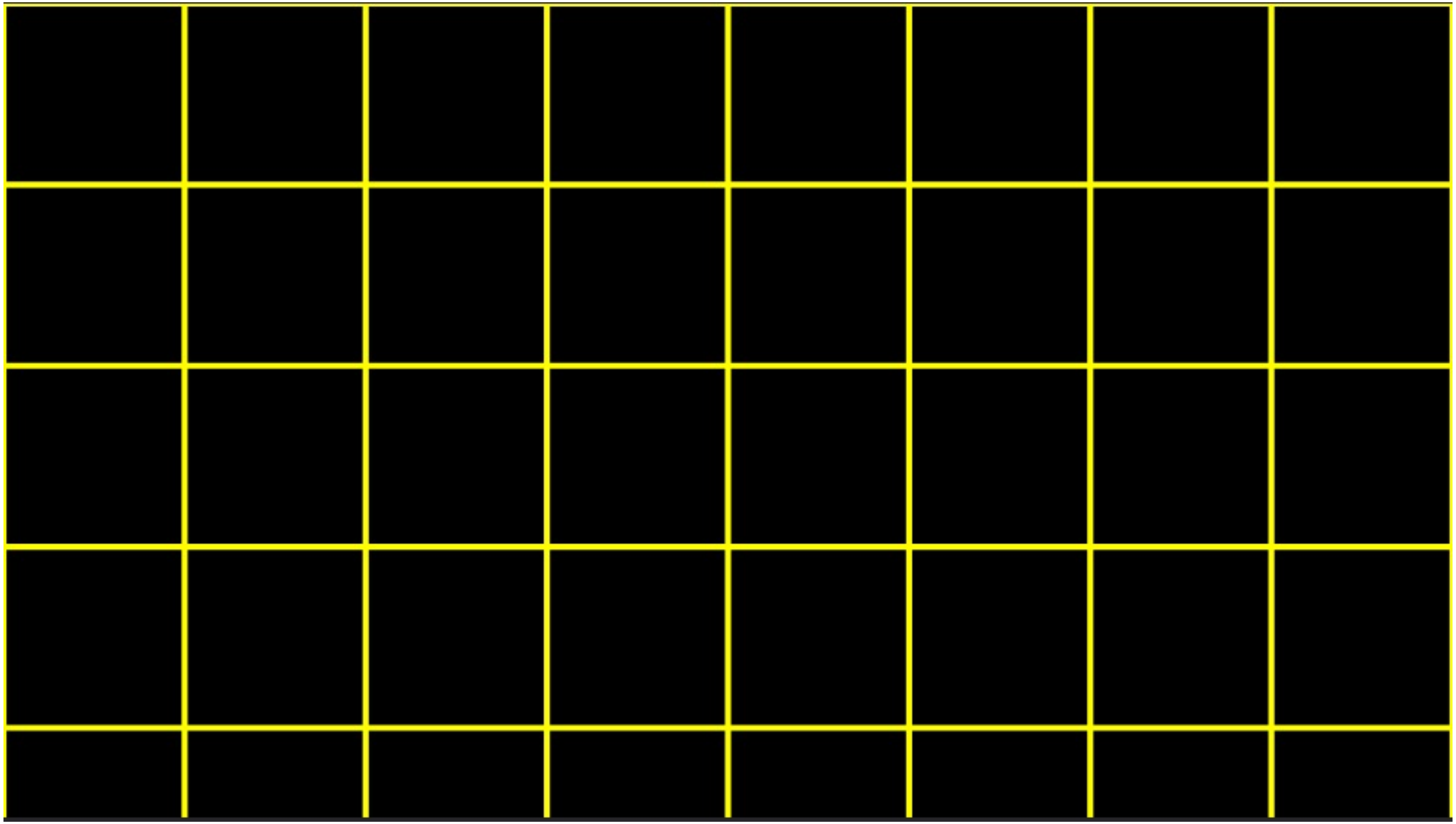
```
59
```

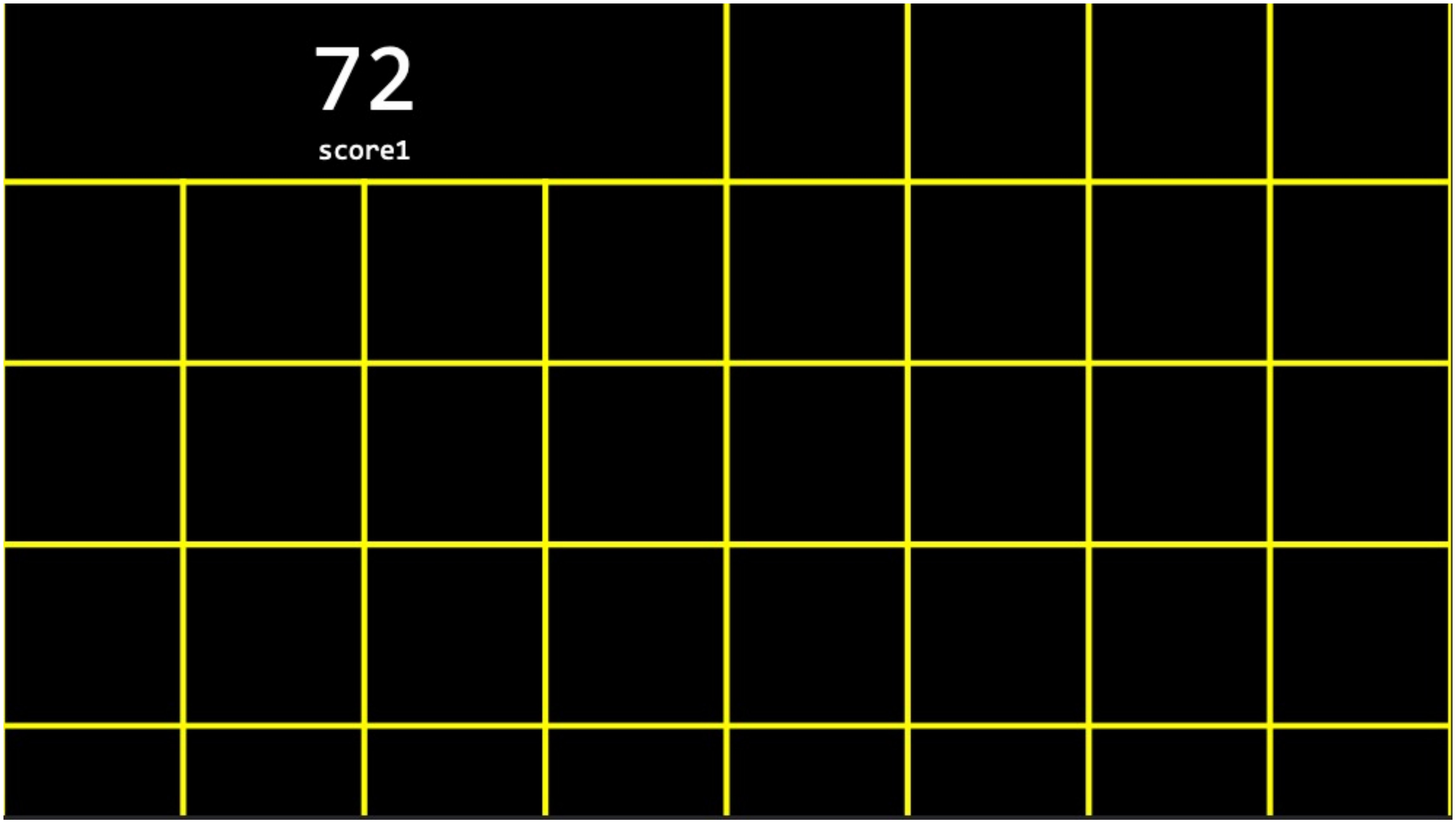
Simple Problem-Solving Tasks

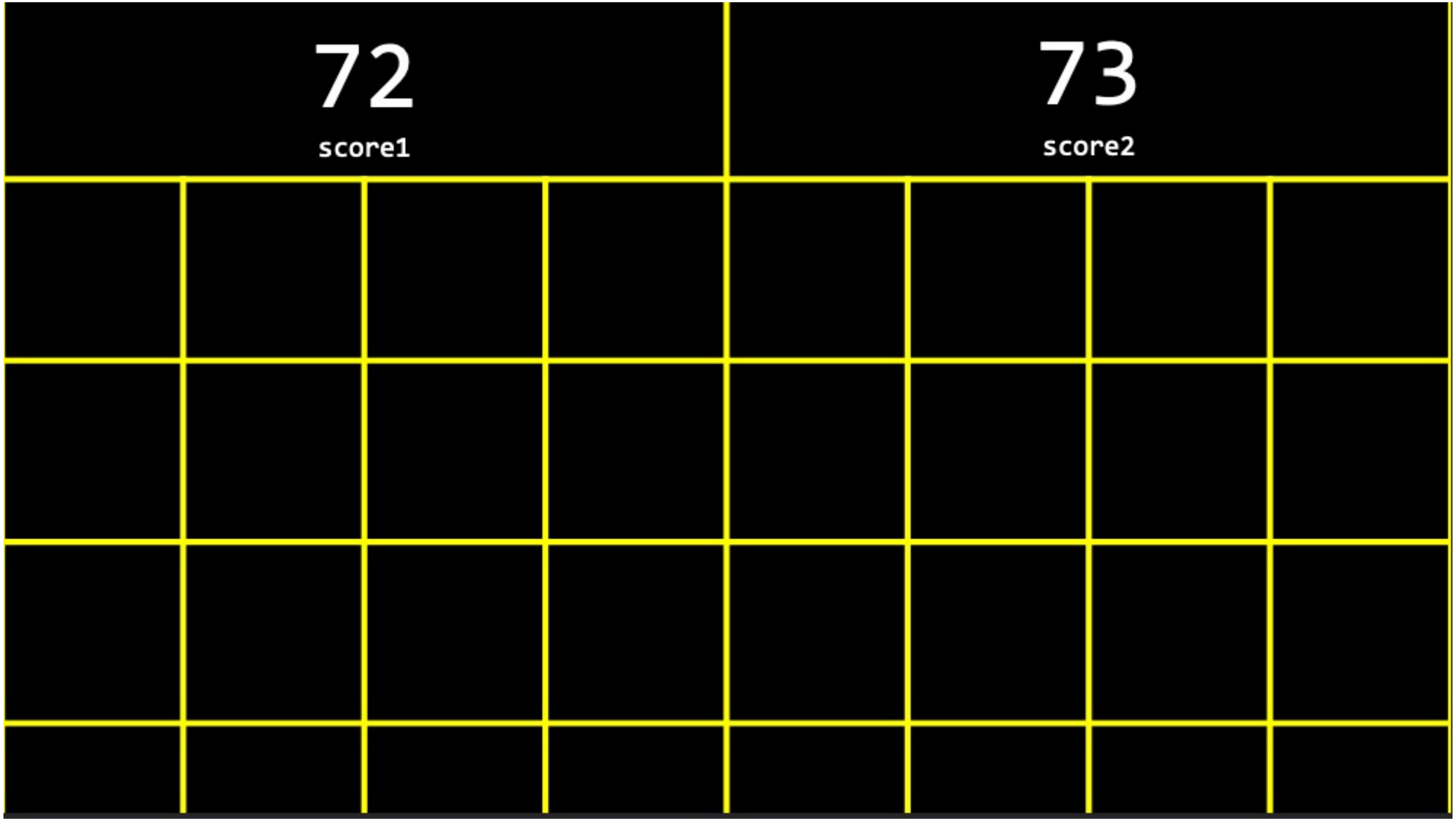
Write a C++ program that will print the average of 3 scores.

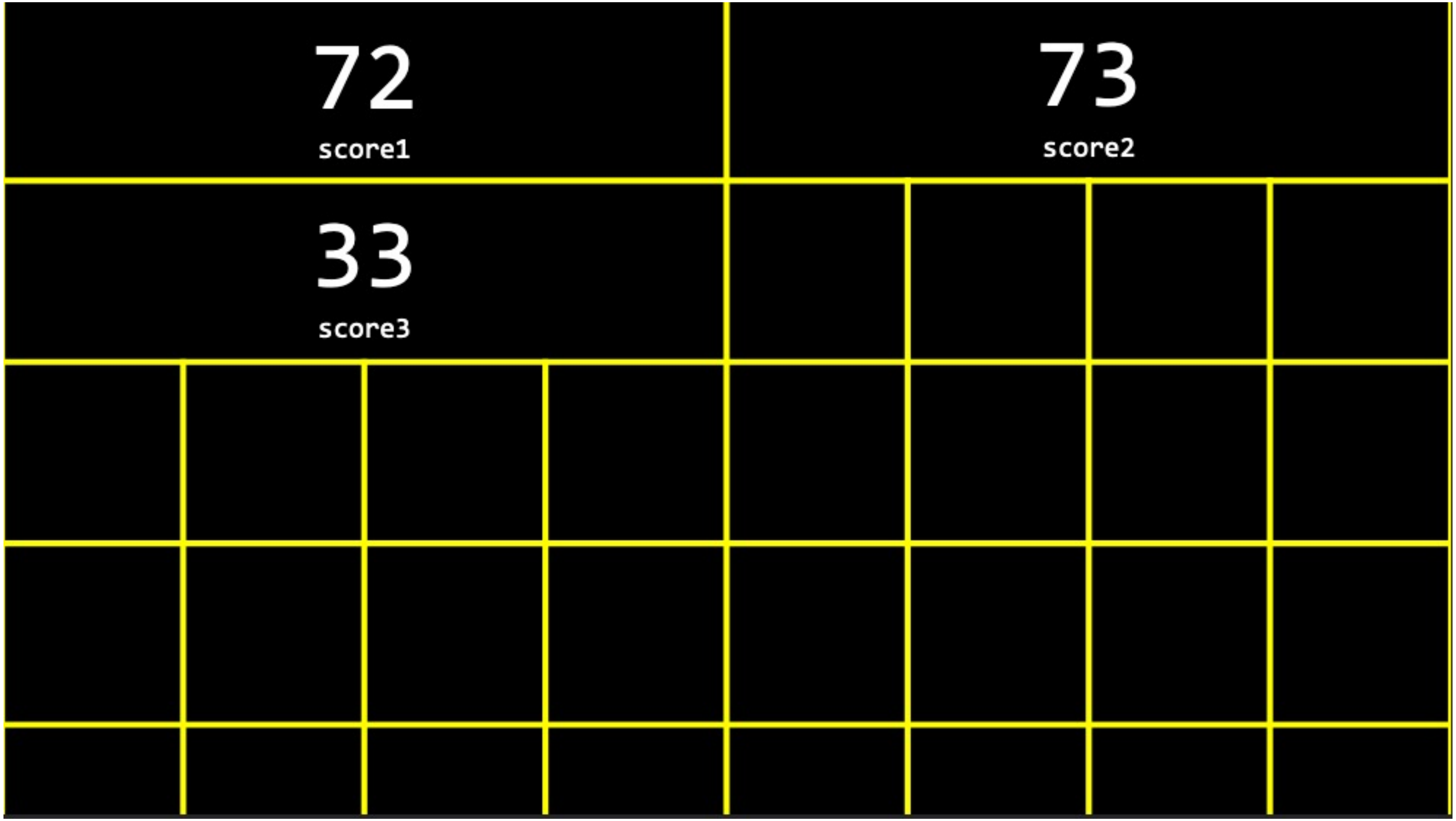
```
#include <iostream>
using namespace std;

int main()
{
        int score1 = 72;
        int score2 = 73;
        int score3 = 33;
        cout << (score1+score2+score3)/3 << endl;
        return 0;
}
```

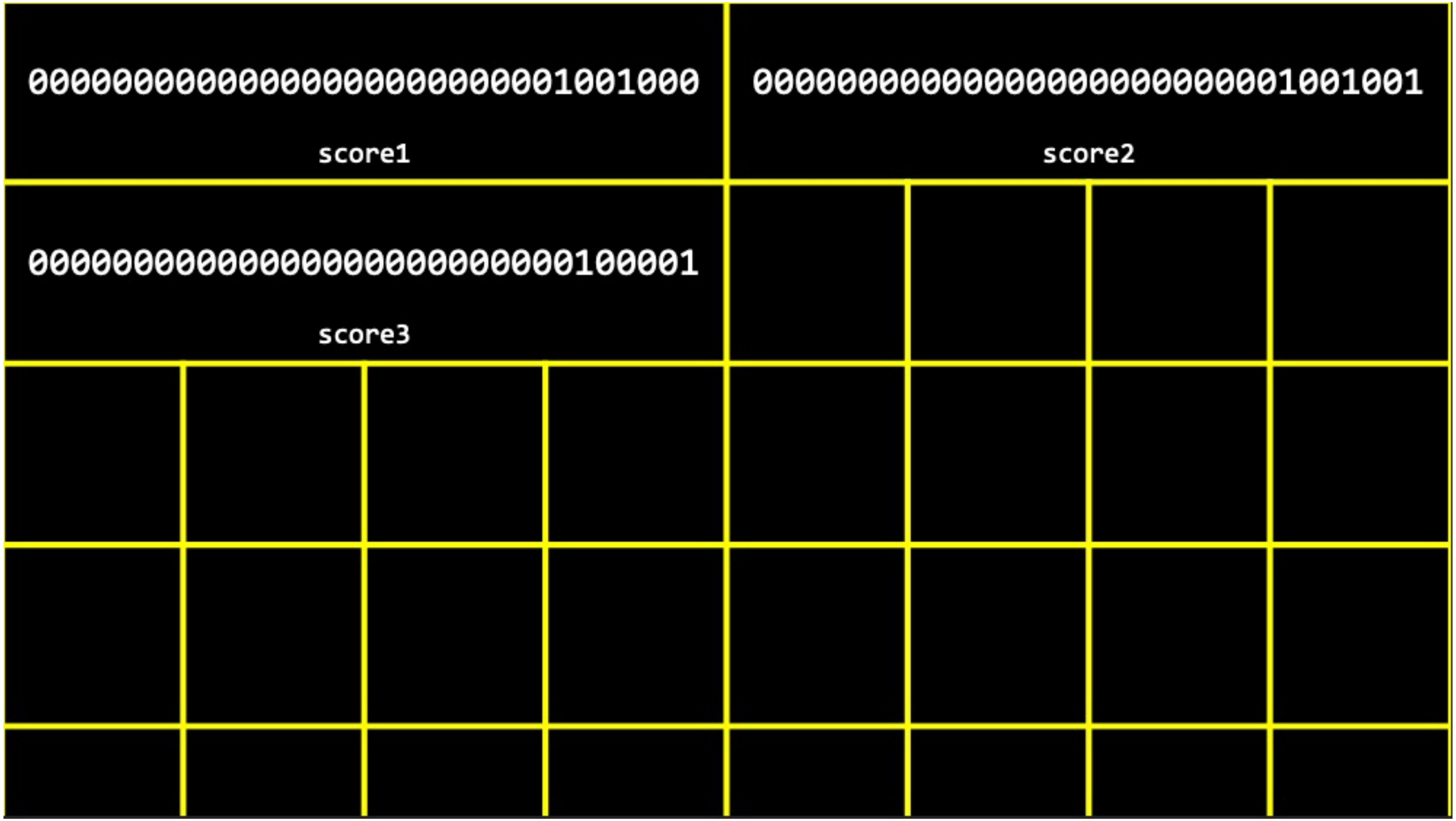
```
59
```




72
score1


72
score1
73
score2


72
score1
73
score2
33
score3

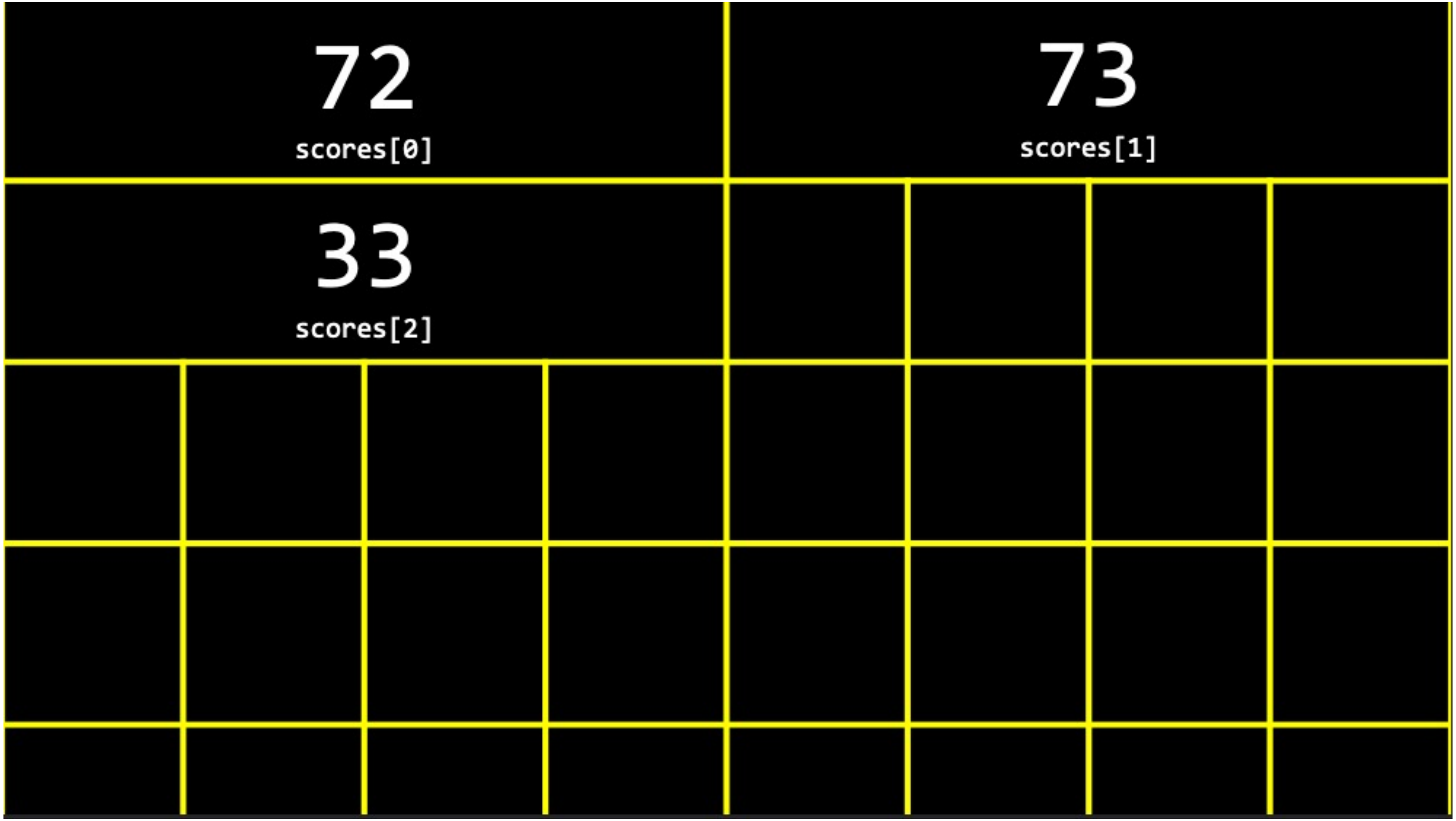

00000000000000000000000001001000
score1
00000000000000000000000001001001
score2
00000000000000000000000000100001
score3

Simple Problem-Solving Tasks

Write a C++ program that will print the average of 3 scores.

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

int main()
{
	// Defining an array of 3 size
	int scores[3];
	scores[0] = 72;    // 0th index
	scores[1] = 73;    // 1st index
	scores[2] = 33;    // 2nd index
	cout << (scores[0]+scores[1]+scores[2])/3 << endl;
	return 0;
}
```

```
59
```


72
scores[0]
73
scores[1]
33
scores[2]

Simple Problem-Solving Tasks

Write a C++ program that will print the average of 3 scores.

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

int main()
{
        // Defining an array of 3 size
        int scores[3];
        scores[0] = 72;    // 0th index
        scores[1] = 73;    // 1st index
        scores[2] = 33;    // 2nd index
        int sum = 0;
        for (int i = 0; i < 3; i++)
                sum += scores[i];
        cout << sum / 3 << endl;
        return 0;
}
```

```
??
```

Simple Problem-Solving Tasks

Write a C++ program that will print the average of 3 scores.

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

int main()
{
	// Defining an array of 3 size
	int scores[3];
	scores[0] = 72;    // 0th index
	scores[1] = 73;    // 1st index
	scores[2] = 33;    // 2nd index
	int sum = 0;
	for (int i = 0; i < 3; i++)
		sum += scores[i];
	cout << sum / 3 << endl;
	return 0;
}
```

```
59
```

Simple Problem-Solving Tasks

Write a C++ program that will print the average of 3 scores.

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

int main()
{
	// Defining an array with the size of 3
	// Initializing it with 3 int values
	int scores[3] = {72, 73, 33};
	int sum = 0;
	for (int i = 0; i < 3; i++)
		sum += scores[i];
	cout << sum / 3 << endl;
	return 0;
}
```

```
59
```

Simple Problem-Solving Tasks

Write a C++ program that will print the average of 3 scores.

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

int main()
{
	// Defining an array with unknown size and
	// Initializing it with 3 int values
	// Thereby, the size will become 3
	int scores[] = {72, 73, 33};
	int sum = 0;
	for (int i = 0; i < 3; i++)
		sum += scores[i];
	cout << sum / 3 << endl;
	return 0;
}
```

```
59
```

Introduction of Arrays (1/2)

- An extremely useful and fundamental data structure
- We use arrays to hold values of **the same datatype** at contiguous memory location.
- One way to analogize the notion of array is to think of your local post office, which usually has a large bank of post office boxes.

Source: https://www.youtube.com/embed/mISkNAfWI8k

Introduction of Arrays (2/2)

- The elements of an array are indexed starting from 0.
- If an array consists of *n* elements, the first element is located at index 0. The last element is located at index (n – 1).

Source: https://www.youtube.com/embed/mlSkNAfWl8k

Array Declaration

```
type name[size];
```

Type: What kind of variable each element of the array will be.
Name: What you want to call your array.
Size: How many elements you would like your array to contain.

Source: https://www.youtube.com/embed/mISkNAfWI8k

Array Declaration

```
double courseScores[4];
```

Type: What kind of variable each element of the array will be.
Name: What you want to call your array.
Size: How many elements you would like your array to contain.

Source: https://www.youtube.com/embed/mISkNAfWI8k

Array Declaration

```
char myCourseGrades[4];
```

Type: What kind of variable each element of the array will be.
Name: What you want to call your array.
Size: How many elements you would like your array to contain.

Source: https://www.youtube.com/embed/mISkNAfWI8k

Array Initialization

```
char myCourseGrades[4] = {‘A’, ‘B’, ‘A’, ‘C’};
```

```
char courseGrades[4];
myCourseGrades[0] = ‘A’;
myCourseGrades[1] = ‘B’;
myCourseGrades[2] = ‘A’;
myCourseGrades[3] = ‘C’;
```

Array Initialization

```
const int SIZE = 4;
char myCourseGrades[SIZE] = {‘A’, ‘B’, ‘A’, ‘C’};
```

```
char courseGrades[4];
myCourseGrades[0] = ‘A’;
myCourseGrades[1] = ‘B’;
myCourseGrades[2] = ‘A’;
myCourseGrades[3] = ‘C’;
```

Array Initialization

```
char myCourseGrades[] = {‘A’, ‘B’, ‘A’, ‘C’};
```

```
char courseGrades[4];
myCourseGrades[0] = ‘A’;
myCourseGrades[1] = ‘B’;
myCourseGrades[2] = ‘A’;
myCourseGrades[3] = ‘C’;
```

Array Initialization

```
char myCourseGrades[] = {‘A’, ‘B’, ‘A’, ‘C’};
```

```
char courseGrades[4];
myCourseGrades[0] = ‘A’;
myCourseGrades[1] = ‘B’;
myCourseGrades[2] = ‘A’;
myCourseGrades[3] = ‘C’;
myCourseGrades[4] = ‘A’;
```

C and C++ are very lenient. It will not prevent you from going “out of bounds” of your array. Be careful!

Array Initialization

```
int x[6] = {19, 10, 8, 17, 9, 15};
```

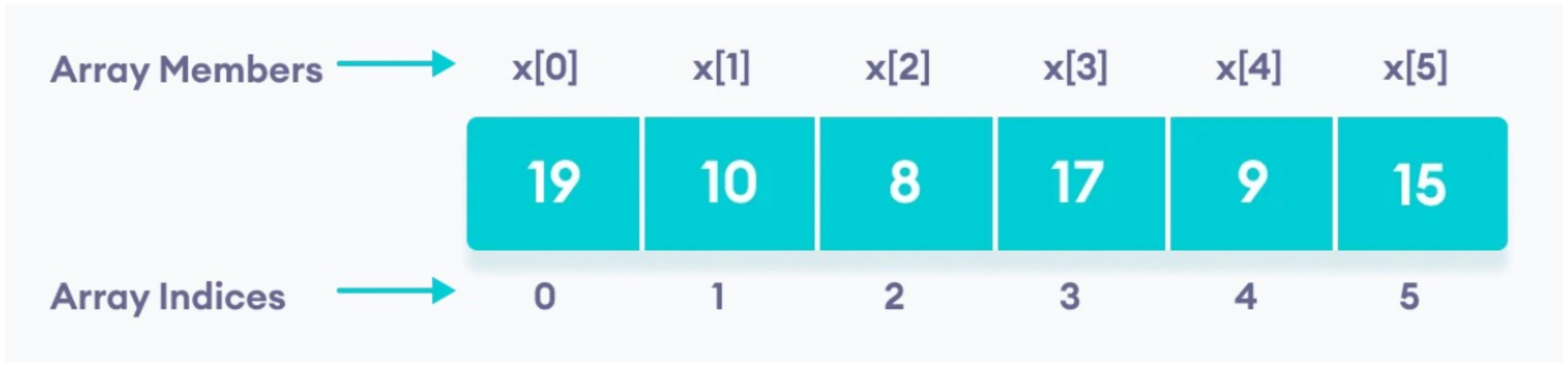
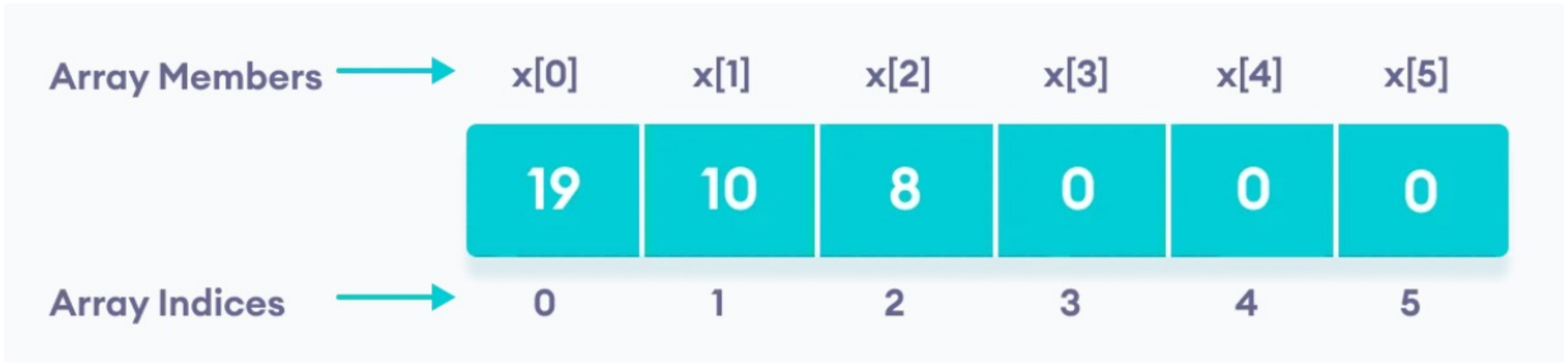


Image Source: Programiz https://www.programiz.com/cpp-programming/arrays

Array Initialization

```
int x[6] = {19, 10, 8};
```



Note: The compiler assigns random values to the remaining places. Oftentimes, this random value is simply 0.

Image Source: Programiz https://www.programiz.com/cpp-programming/arrays

Array Initialization

```
int x[6] = {19, 10, 8};
x[3] = 17;
x[4] = 9;
x[5] = 15;
```

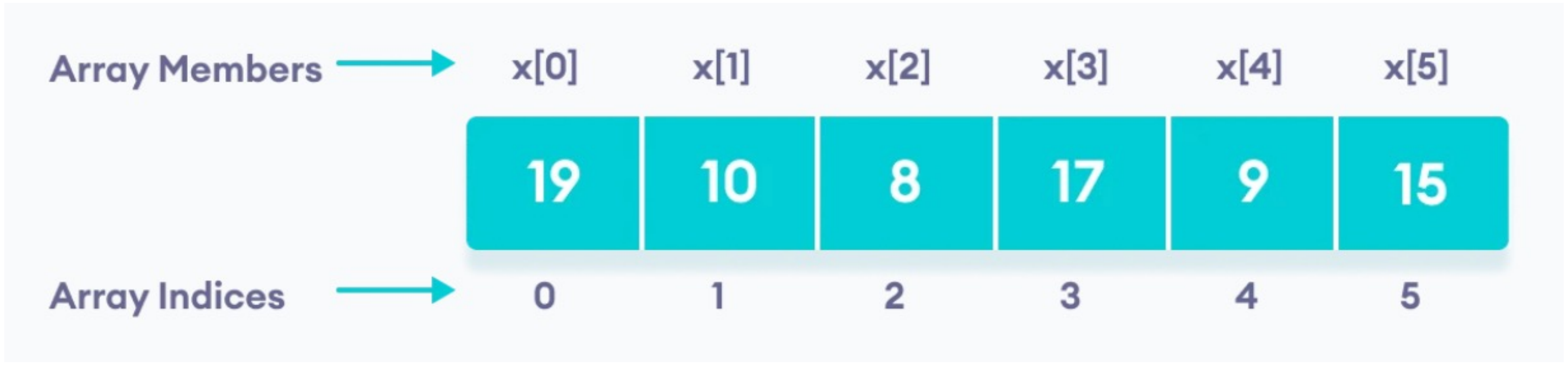


Image Source: Programiz https://www.programiz.com/cpp-programming/arrays

Output Tracing

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

int main()
{
        // Declare an array of size 10
        int myArray[10];

        // Initializing the array with int values
        for (int i = 0; i < 10; i++)
                myArray[i] = i;

        // Displaying the elements of the array
        for (int i = 0; i < 10; i++)
                cout << myArray[i] << " ";
        return 0;
}
```

```
??
```

Output Tracing

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

int main()
{
        // Declare an array of size 10
        int myArray[10];

        // Initializing the array with int values
        for (int i = 0; i < 10; i++)
                myArray[i] = i;

        // Displaying the elements of the array
        for (int i = 0; i < 10; i++)
                cout << myArray[i] << “ ”;
        return 0;
}
```

```
0 1 2 3 4 5 6 7 8 9
```

Output Tracing

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

int main()
{
        // Declare an array of size 10
        int myArray[10];

        // Initializing the array with int values
        for (int i = 0; i < 10; i++)
                myArray[i] = i;

        for (int i = 0; i < 10; i = i + 2)
                cout << myArray[i] << " ";
        return 0;
}
```

```
??
```

Output Tracing

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

int main()
{
	// Declare an array of size 10
	int myArray[10];

	// Initializing the array with int values
	for (int i = 0; i < 10; i++)
		myArray[i] = i;

	for (int i = 0; i < 10; i = i + 2)
		cout << myArray[i] << “ ”;
	return 0;
}
```

```
0 2 4 6 8
```

Output Tracing

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

int main()
{
        // Declare an array of size 10
        int myArray[10];

        // Initializing the array with int values
        for (int i = 0; i < 10; i++)
                myArray[i] = i;

        for (int i = 1; i < 10; i = i + 2)
                cout << myArray[i] << “ ”;
        return 0;
}
```

```
??
```

Output Tracing

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

int main()
{
        // Declare an array of size 10
        int myArray[10];

        // Initializing the array with int values
        for (int i = 0; i < 10; i++)
                myArray[i] = i;

        for (int i = 1; i < 10; i = i + 2)
                cout << myArray[i] << " ";
        return 0;
}
```

```
1 3 5 7 9
```

Output Tracing

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

int main()
{
	// Declare an array of size 10
	int myArray[10];

	for (int i = 0; i < 10; i++)
		myArray[i] = i + 1;

	unsigned int sum = 0;
	for (int i = 0; i < 10; i++)
		sum += myArray[i];

	cout << sum << endl;
	return 0;
}
```

```
??
```

Output Tracing

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

int main()
{
        // Declare an array of size 10
        int myArray[10];

        for (int i = 0; i < 10; i++)
                myArray[i] = i + 1;

        unsigned int sum = 0;
        for (int i = 0; i < 10; i++)
                sum += myArray[i];

        cout << sum << endl;
        return 0;
}
```

```
55
```

Output Tracing

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

int main()
{
	// Declare an array of size 10
	int myArray[10];

	for (int i = 0; i < 10; i++)
		myArray[i] = 5 * (i + 1);

	for (int i = 0; i < 10; i++)
		cout << myArray[i] << endl;

	return 0;
}
```

```
??
```

Output Tracing

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

int main()
{
        // Declare an array of size 10
        int myArray[10];

        for (int i = 0; i < 10; i++)
                myArray[i] = 5 * (i + 1);

        for (int i = 0; i < 10; i++)
                cout << myArray[i] << endl;

        return 0;
}
```

```
5 10 15 20 25 30 35 40 45 50
```

Sample Program

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

int main()
{
        // Declare an array of size 10
        int myArray[4];
        cout << “Please enter 4 integers: ” << endl;
        for (int i = 0; i < 4; i++)
                cin >> myArray[i];

        cout << “Values in array are now: ” << endl;
        for (int i = 0; i < 4; i++)
                cout << myArray[i] << “ ”;

        cout << endl;
        return 0;
}
```

Note: We can treat individual element of arrays as variables and use it as we have done before.

```
Please enter 4 integers:
_
```

Sample Program

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

int main()
{
        // Declare an array of size 10
        int myArray[4];
        cout << “Please enter 4 integers: ” << endl;
        for (int i = 0; i < 4; i++)
                cin >> myArray[i];

        cout << “Values in array are now: ” << endl;
        for (int i = 0; i < 4; i++)
                cout << myArray[i] << “ ”;

        cout << endl;
        return 0;
}
```

```
Please enter 4 integers:
40
_
```

Sample Program

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

int main()
{
        // Declare an array of size 10
        int myArray[4];
        cout << “Please enter 4 integers: ” << endl;
        for (int i = 0; i < 4; i++)
                cin >> myArray[i];

        cout << “Values in array are now: ” << endl;
        for (int i = 0; i < 4; i++)
                cout << myArray[i] << “ ”;

        cout << endl;
        return 0;
}
```

```
Please enter 4 integers:
40
45
_
```

Sample Program

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

int main()
{
        // Declare an array of size 10
        int myArray[4];
        cout << “Please enter 4 integers: ” << endl;
        for (int i = 0; i < 4; i++)
                cin >> myArray[i];

        cout << “Values in array are now: ” << endl;
        for (int i = 0; i < 4; i++)
                cout << myArray[i] << “ ”;

        cout << endl;
        return 0;
}
```

```
Please enter 4 integers:
40
45
30
_
```

Sample Program

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

int main()
{
        // Declare an array of size 10
        int myArray[4];
        cout << “Please enter 4 integers: ” << endl;
        for (int i = 0; i < 4; i++)
                cin >> myArray[i];

        cout << “Values in array are now: ” << endl;
        for (int i = 0; i < 4; i++)
                cout << myArray[i] << “ ”;

        cout << endl;
        return 0;
}
```

```
Please enter 4 integers:
40
45
30
60
Values in array are now: 40 45 30 60
```

Sample Program

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

int main()
{
        // Declare and initialize an array of size 10
        int foo[4] = {2, 5, 1, 10};
        int bar[4] = {2, 5, 1, 10};

        // Trying to copy foo to bar
        bar = foo;

        for (int i = 0; i < 4; i++)
                cout << bar[i] << " ";

        cout << endl;
        return 0;
}
```

Note: We cannot assign one array to another sing the assignment operator.

```
Error! This is not legal in C or C++.
```

Sample Program

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

int main()
{
        // Declare and initialize an array of size 10
        int foo[4] = {2, 5, 1, 10};
        int bar[4] = {2, 5, 1, 10};

        // Successful copy of foo to bar
        for (int i = 0; i < 4: i++)
                bar[i] = foo[i];

        for (int i = 0; i < 4; i++)
                cout << bar[i] << " ";
        cout << endl;
        return 0;
}
```

Note: We must use loop to copy over the elements of array one at a time.

```
2 5 1 10
```

Multi-dimensional Arrays

- Arrays can consist of more than a single dimension.

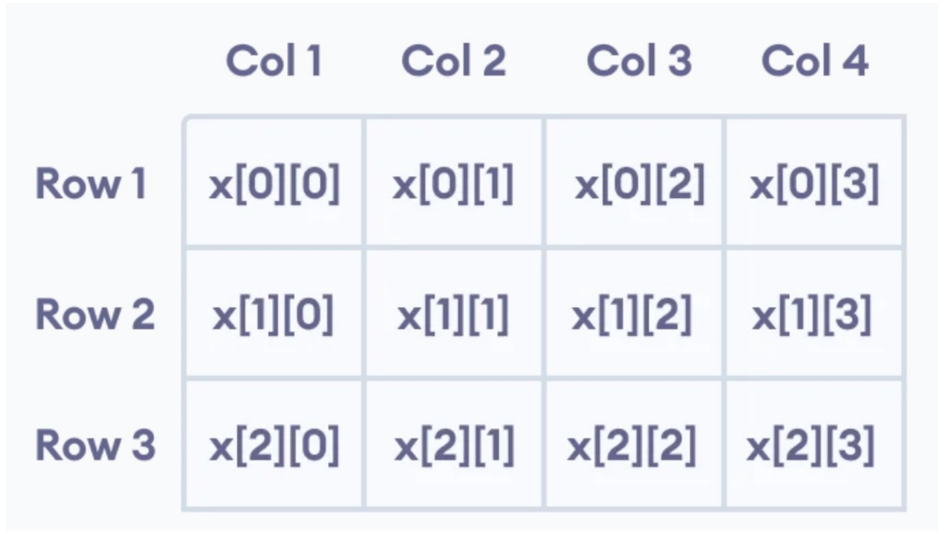
```
bool TicTacToe[3][3];
```

- You can choose to think of this as a 3*3 grid of cells
  - In memory though, it is really just a 9-element one-dimensional array.
  - Multi-dimensional arrays are great **abstraction** to help visualize complex representation.

Source: https://www.youtube.com/embed/mISkNAfWI8k

Two-dimensional Arrays Declaration

```
int x[3][4];
```

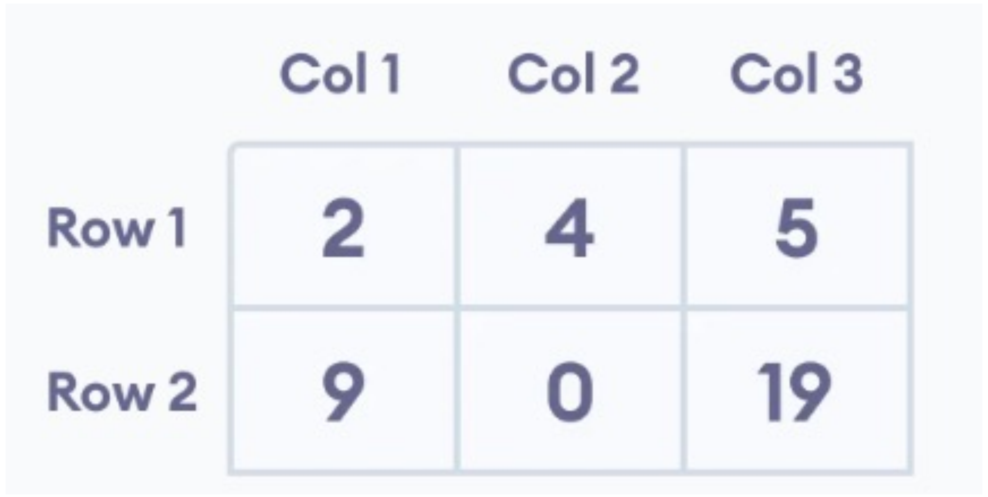


	Col 1	Col 2	Col 3	Col 4
Row 1	x[0][0]	x[0][1]	x[0][2]	x[0][3]
Row 2	x[1][0]	x[1][1]	x[1][2]	x[1][3]
Row 3	x[2][0]	x[2][1]	x[2][2]	x[2][3]

Image Source: Programiz https://www.programiz.com/cpp-programming/multidimensional-arrays

Two-dimensional Arrays Initialization

```
int x[2][3] = {{2,4,5} , {9,0,19}};
```

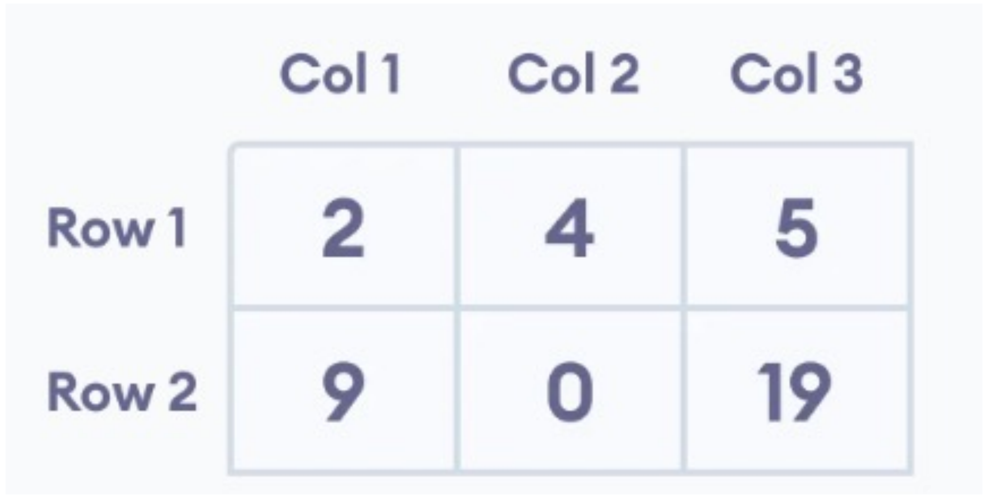


	Col 1	Col 2	Col 3
Row 1	2	4	5
Row 2	9	0	19

Image Source: Programiz https://www.programiz.com/cpp-programming/multidimensional-arrays

Two-dimensional Arrays Initialization

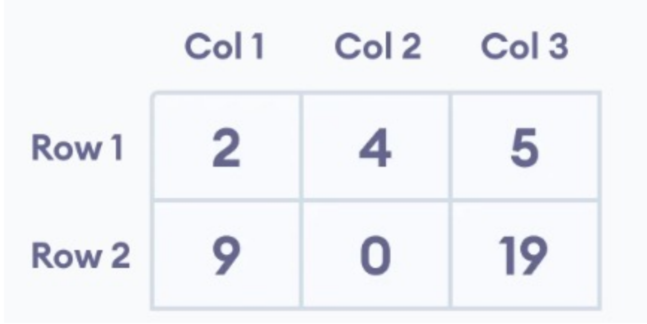
```
int x[2][3] = {2,4,5,9,0,19};
```



	Col 1	Col 2	Col 3
Row 1	2	4	5
Row 2	9	0	19

Image Source: Programiz https://www.programiz.com/cpp-programming/multidimensional-arrays

Output Tracing

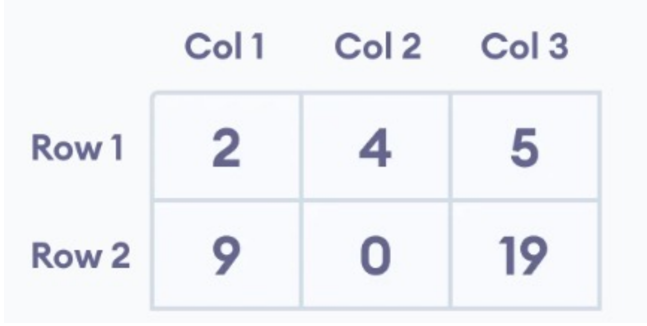


	Col 1	Col 2	Col 3
Row 1	2	4	5
Row 2	9	0	19

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

int main()
{
        // Declare and initialize a two-dimensional array of size 6 (2*3)
        int myTwoDimArray[2][3] = {{2,4,5},{9,0,19}};

        for (int r = 0; r < 2; r++)
        {
                for(int c = 0; c < 3; c++)
                {
                        cout << myTwoDimArray[r][c] << " ";
                }
                cout << endl;
        }
        return 0;
}
```

```
??
```

Output Tracing

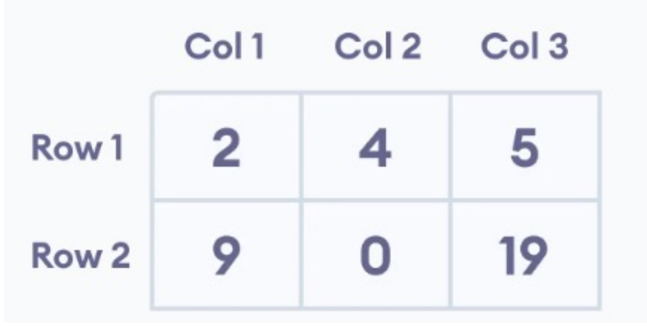


	Col 1	Col 2	Col 3
Row 1	2	4	5
Row 2	9	0	19

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

int main()
{
        // Declare and initialize a two-dimensional array of size 6 (2*3)
        int myTwoDimArray[2][3] = {{2,4,5},{9,0,19}};

        for (int r = 0; r < 2; r++)
        {
                for(int c = 0; c < 3; c++)
                {
                        cout << myTwoDimArray[r][c] << " ";
                }
                cout << endl;
        }
        return 0;
}
```

```
2 4 5
9 0 19
```

Output Tracing

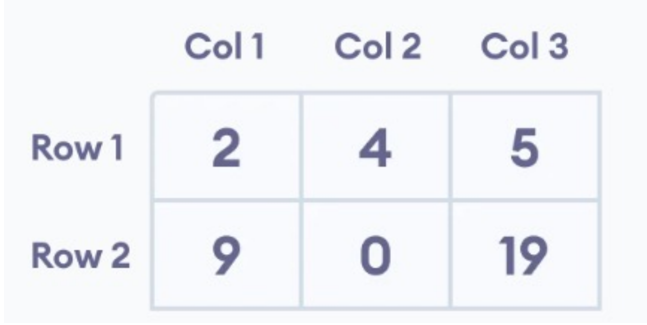


	Col 1	Col 2	Col 3
Row 1	2	4	5
Row 2	9	0	19

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

int main()
{
        // Declare and initialize a two-dimensional array of size 6 (2*3)
        int myTwoDimArray[2][3] = {{2,4,5},{9,0,19}};

        for (int r = 0; r < 2; r++)
        {
                for(int c = 2; c >= 0; c--)
                {
                        cout << myTwoDimArray[r][c] << “ ”;
                }
                cout << endl;
        }
        return 0;
}
```

```
??
```

Output Tracing



	Col 1	Col 2	Col 3
Row 1	2	4	5
Row 2	9	0	19

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

int main()
{
        // Declare and initialize a two-dimensional array of size 6 (2*3)
        int myTwoDimArray[2][3] = {{2,4,5},{9,0,19}};

        for (int r = 0; r < 2; r++)
        {
                for(int c = 2; c >= 0; c--)
                {
                        cout << myTwoDimArray[r][c] << “ ”;
                }
                cout << endl;
        }
        return 0;
}
```

```
5 4 2
19 0 9
```

What’s in the Two-Dim Array?

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

int main()
{
	// Declare a two-dimensional array of size 20 (2*10)
	int myTwoDimArray[2][10];

	for (int r = 0; r < 2; r++)
	{
		for(int c = 0; c < 10; c++)
		{
			myTwoDimArray[r][c] = (r+1) * (c+1);
		}
	}
	return 0;
}
```

myTwoDimArray

??

What’s in the Two-Dim Array?

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

int main()
{
        // Declare a two-dimensional array of size 20 (2*10)
        int myTwoDimArray[2][10];

        for (int r = 0; r < 2; r++)
        {
                for(int c = 0; c < 10; c++)
                {
                        myTwoDimArray[r][c] = (r+1) * (c+1);
                }
        }
        return 0;
}
```

myTwoDimArray

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
```

What's in the Two-Dim Array?

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

int main()
{
        // Declare a two-dimensional array of size 50 (5*10)
        int myTwoDimArray[5][10];

        for (int r = 0; r < 5; r++)
        {
                for(int c = 0; c < 10; c++)
                {
                        myTwoDimArray[r][c] = (r+1) * (c+1);
                }
        }
        return 0;
}
```

myTwoDimArray

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
```

myTwoDimArray

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
```

Output Tracing

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

int main()
{
        // Declare a two-dimensional array of size 50 (5*10)
        int myTwoDimArray[5][10];
        for (int r = 0; r < 5; r++)
                for(int c = 0; c < 10; c++)
                        myTwoDimArray[r][c] = (r+1) * (c+1);

        for (int i = 0; i < 5; i++)
        {
                int sum = 0;
                for(int j = 0; j < 10; j++)
                {
                        sum += myTwoDimArray[i][j];
                }
                cout << sum << endl;
        }
}
```

```
??
```

Output Tracing

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

int main()
{
	// Declare a two-dimensional array of size 50 (5*10)
	int myTwoDimArray[5][10];
	for (int r = 0; r < 5; r++)
		for(int c = 0; c < 10; c++)
			myTwoDimArray[r][c] = (r+1) * (c+1);

	for (int i = 0; i < 5; i++)
	{
		int sum = 0;
		for(int j = 0; j < 10; j++)
		{
			sum += myTwoDimArray[i][j];
		}
		cout << sum << endl;
	}
}
```

myTwoDimArray

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
```

```
55
110
165
220
275
```

myTwoDimArray

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
```

Output Tracing

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

int main()
{
        // Declare a two-dimensional array of size 50 (5*10)
        int myTwoDimArray[5][10];
        for (int r = 0; r < 5; r++)
                for(int c = 0; c < 10; c++)
                        myTwoDimArray[r][c] = (r+1) * (c+1);

        for (int i = 0; i < 10; i++)
        {
                int sum = 0;
                for(int j = 0; j < 5; j++)
                {
                        sum += myTwoDimArray[j][i];
                }
                cout << sum << endl;
        }
}
```

```
??
```

Output Tracing

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

int main()
{
	// Declare a two-dimensional array of size 50 (5*10)
	int myTwoDimArray[5][10];
	for (int r = 0; r < 5; r++)
		for(int c = 0; c < 10; c++)
			myTwoDimArray[r][c] = (r+1) * (c+1);

	for (int i = 0; i < 10; i++)
	{
		int sum = 0;
		for(int j = 0; j < 5; j++)
		{
			sum += myTwoDimArray[j][i];
		}
		cout << sum << endl;
	}
}
```

myTwoDimArray

```
1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
```

```
15
30
45
60
75
90
105
120
135
150
```

Arrays and Functions

- Arrays can be passed as arguments to a functions. When declaring the function, simply specify the array as a parameter (w/o the size).
- It is important to note that arrays are ***passed by reference***. Any changes made to the array within the function will be observed in the calling scope.

Source: MIT OCW 6.096 Intro to C++ https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-096-introduction-to-c-january-iap-2011/lecture-notes/MIT6_096IAP11_lec04.pdf

Sample Program

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

void ModifyArray(int myArray[], int arrayLength)
{
        for(int i = 0; i < arrayLength; i++)
                myArray[i] = myArray[i] * myArray[i];
}

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};

        ModifyArray(foo, 5);
        for (int i = 0; i < 5; i++)
                cout << foo[i] << " ";
}
```

```
??
```

Sample Program

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

void ModifyArray(int myArray[], int arrayLength)
{
        for(int i = 0; i < arrayLength; i++)
                myArray[i] = myArray[i] * myArray[i];
}

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};

        ModifyArray(foo, 5);
        for (int i = 0; i < 5; i++)
                cout << foo[i] << " ";
}
```

```
1 4 9 16 25
```

Sample Program

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

void ModifyArray(int myArray[5])
{
        for(int i = 0; i < 5; i++)
                myArray[i] = myArray[i] * myArray[i];
}

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};

        ModifyArray(foo);
        for (int i = 0; i < 5; i++)
                cout << foo[i] << “ ”;
}
```

```
??
```

Sample Program

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

void ModifyArray(int myArray[5])
{
        for(int i = 0; i < 5; i++)
                myArray[i] = myArray[i] * myArray[i];
}

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};

        ModifyArray(foo);
        for (int i = 0; i < 5; i++)
                cout << foo[i] << " ";
}
```

Note: Although this will still work, the size of the array inside the function scope is set to 5.

```
1 4 9 16 25
```

Sample Program

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

void ModifyArray(int myArray[5])
{
        for(int i = 0; i < 5; i++)
                myArray[i] = myArray[i] * myArray[i];
}

int main()
{
        // Declare and initialize an array of size 5
        int foo[3] = {1, 2, 3};

        ModifyArray(foo);
        for (int i = 0; i < 3; i++)
                cout << foo[i] << " ";
}
```

```
??
```

Sample Program

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

void ModifyArray(int myArray[5])
{
        for(int i = 0; i < 5; i++)
                myArray[i] = myArray[i] * myArray[i];
}

int main()
{
        // Declare and initialize an array of size 5
        int foo[3] = {1, 2, 3};

        ModifyArray(foo);
        for (int i = 0; i < 3; i++)
                cout << foo[i] << “ ”;
}
```

Note: Not an error! The contiguous memory locations’ values are changed.

```
1 4 9
```

Output Tracing

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

int sum(int myArray[], int arrayLength)
{
        int sum = 0;
        for(int i = 0; i < arrayLength; i++)
                sum += myArray[i];
        return sum;
}

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};

        cout << sum(foo, 5) << endl;
        return 0;
}
```

```
??
```

Output Tracing

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

int sum(int myArray[], int arrayLength)
{
        int sum = 0;
        for(int i = 0; i < arrayLength; i++)
                sum += myArray[i];
        return sum;
}

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};

        cout << sum(foo, 5) << endl;
        return 0;
}
```

```
15
```

Simple Program

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

int main()
{
	int arr[5] = {10, 20, 30, 40, 50};
	cout << arr[0] << endl;
	cout << arr[1] << endl;
	cout << arr[2] << endl;
	cout << arr[3] << endl;
	cout << arr[4] << endl;
	return 0;
}
```

```
??
```

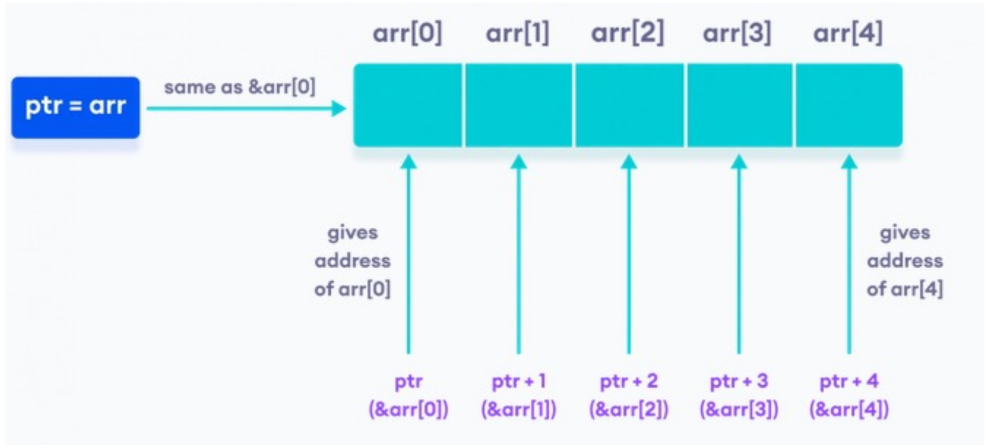
Simple Program

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

int main()
{
	int arr[5] = {10, 20, 30, 40, 50};
	cout << arr[0] << endl;
	cout << arr[1] << endl;
	cout << arr[2] << endl;
	cout << arr[3] << endl;
	cout << arr[4] << endl;
	return 0;
}
```

```
10
20
30
40
50
```

Simple Program

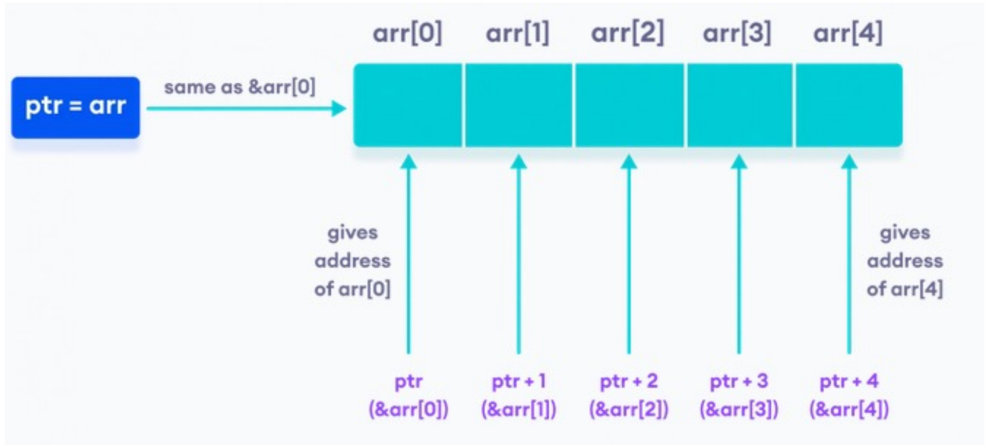
```
#include <iostream>
using namespace std;

int main()
{
        int arr[5] = {10, 20, 30, 40, 50};
        // A pointer variable that is pointing at the
        // first element of the array
        int *ptr = &arr[0];

        cout << *ptr << endl;
        cout << *(ptr+1) << endl;
        cout << *(ptr+2) << endl;
        cout << *(ptr+3) << endl;
        cout << *(ptr+4) << endl;
        return 0;
}
```



```
10
20
30
40
50
```

Simple Program

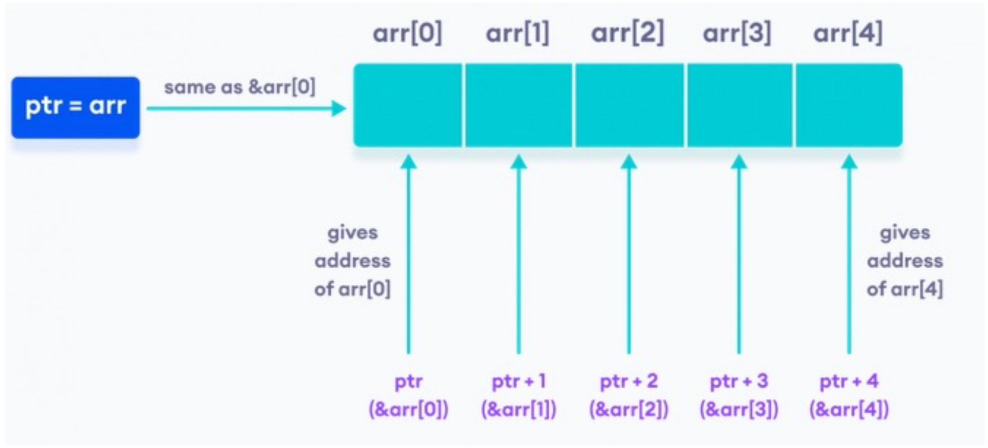
```
#include <iostream>
using namespace std;

int main()
{
        int arr[5] = {10, 20, 30, 40, 50};
        // A pointer variable that is pointing at the
        // first element of the array
        int *ptr = arr;

        cout << *ptr << endl;
        cout << *(ptr+1) << endl;
        cout << *(ptr+2) << endl;
        cout << *(ptr+3) << endl;
        cout << *(ptr+4) << endl;
        return 0;
}
```



```
10
20
30
40
50
```

Simple Program

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

int main()
{
        int arr[5] = {10, 20, 30, 40, 50};
        // A pointer variable that is pointing at the
        // first element of the array
        int *ptr = arr;

        cout << *(ptr+0) << endl;
        cout << *(ptr+1) << endl;
        cout << *(ptr+2) << endl;
        cout << *(ptr+3) << endl;
        cout << *(ptr+4) << endl;
        return 0;
}
```



```
10
20
30
40
50
```

Arrays and Pointers

- The name of an array is actually a pointer to the first element in the array.
- Writing `myArray[3]` tells the compiler to return the element that is 3 away from the starting element of `myArray`.
- This explains why arrays are always passed by reference: passing an array means passing a pointer.
- This also explains why array indices start at 0: the first element of an array is the element that is 0 away from the array.

Source: MIT OCW 6.096 Intro to C++ https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-096-introduction-to-c-january-iap-2011/lecture-notes/MIT6_096IAP11_lec05.pdf

Output Tracing

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

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};
        int *p = foo;
        for (int i = 0; i < 5; i++)
                cout << *(p + i) << " ";
}
```

```
??
```

Output Tracing

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

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};
        int *p = foo;
        for (int i = 0; i < 5; i++)
                cout << *(p + i) << " ";
}
```

```
1 2 3 4 5
```

Output Tracing

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

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};
        int *p = foo;
        for (int i = 0; i < 5; i++)
                cout << (p + i) << endl;
}
```

```
??
```

Output Tracing

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

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};
        int *p = foo;
        for (int i = 0; i < 5; i++)
                cout << (p + i) << endl;
}
```

```
0x7ffde58866f0
0x7ffde58866f4
0x7ffde58866f8
0x7ffde58866fc
0x7ffde5886700
```

Note: Accessing the addresses of the elements of the array using pointer.

Output Tracing

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

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};
        for (int i = 0; i < 5; i++)
                cout << &foo[i] << endl;
}
```

```
??
```

Output Tracing

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

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};
        for (int i = 0; i < 5; i++)
                cout << &foo[i] << endl;
}
```

```
0x7ffde58866f0
0x7ffde58866f4
0x7ffde58866f8
0x7ffde58866fc
0x7ffde5886700
```

Note: Accessing the addresses of the elements of the array.

Output Tracing

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

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};
        for (int i = 0; i < 5; i++)
                cout << (foo + i) << endl;
}
```

```
??
```

Output Tracing

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

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};
        for (int i = 0; i < 5; i++)
                cout << (foo + i) << endl;
}
```

```
0x7ffde58866f0
0x7ffde58866f4
0x7ffde58866f8
0x7ffde58866fc
0x7ffde5886700
```

Note: Accessing the addresses of the contiguous memory locations of the array.

Output Tracing

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

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};
        for (int i = 0; i < 5; i++)
                cout << *(foo + i) << endl;
}
```

```
??
```

Output Tracing

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

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};
        for (int i = 0; i < 5; i++)
                cout << *(foo + i) << endl;
}
```

```
1
2
3
4
5
```

Note: * operator is going to de-reference and display the value of that memory address.
`foo[i]` and `*(foo + i)` are equivalent statements.

Output Tracing

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

int sum(int *arr, int arrayLength)
{
        int sum = 0;
        for(int i = 0; i < arrayLength; i++)
                sum += *(arr + i);
        return sum;
}

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};

        cout << sum(foo, 5) << endl;
        return 0;
}
```

```
??
```

Output Tracing

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

int sum(int *arr, int arrayLength)
{
        int sum = 0;
        for(int i = 0; i < arrayLength; i++)
                sum += *(arr + i);
        return sum;
}

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};

        cout << sum(foo, 5) << endl;
        return 0;
}
```

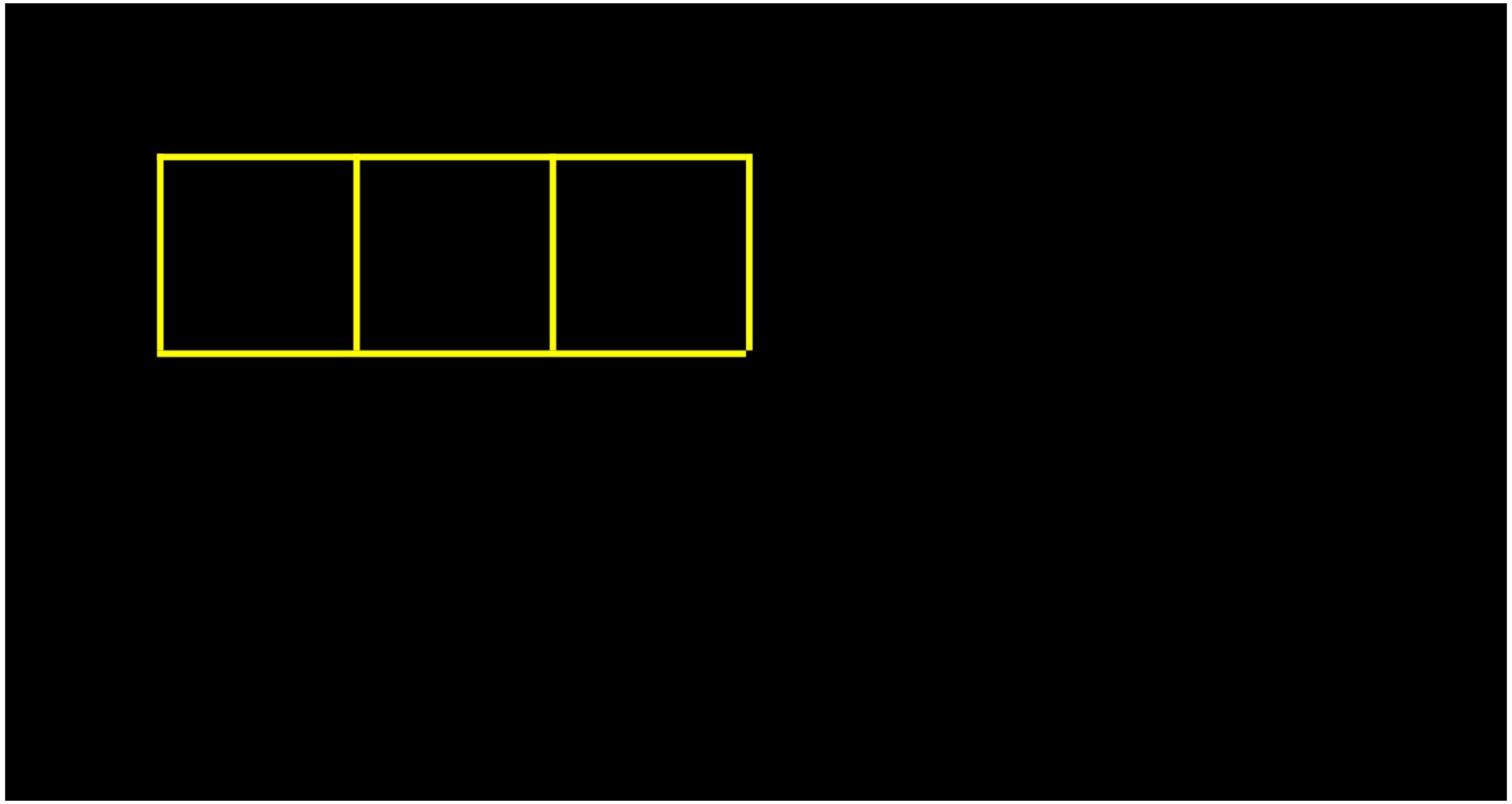
```
15
```

Output Tracing

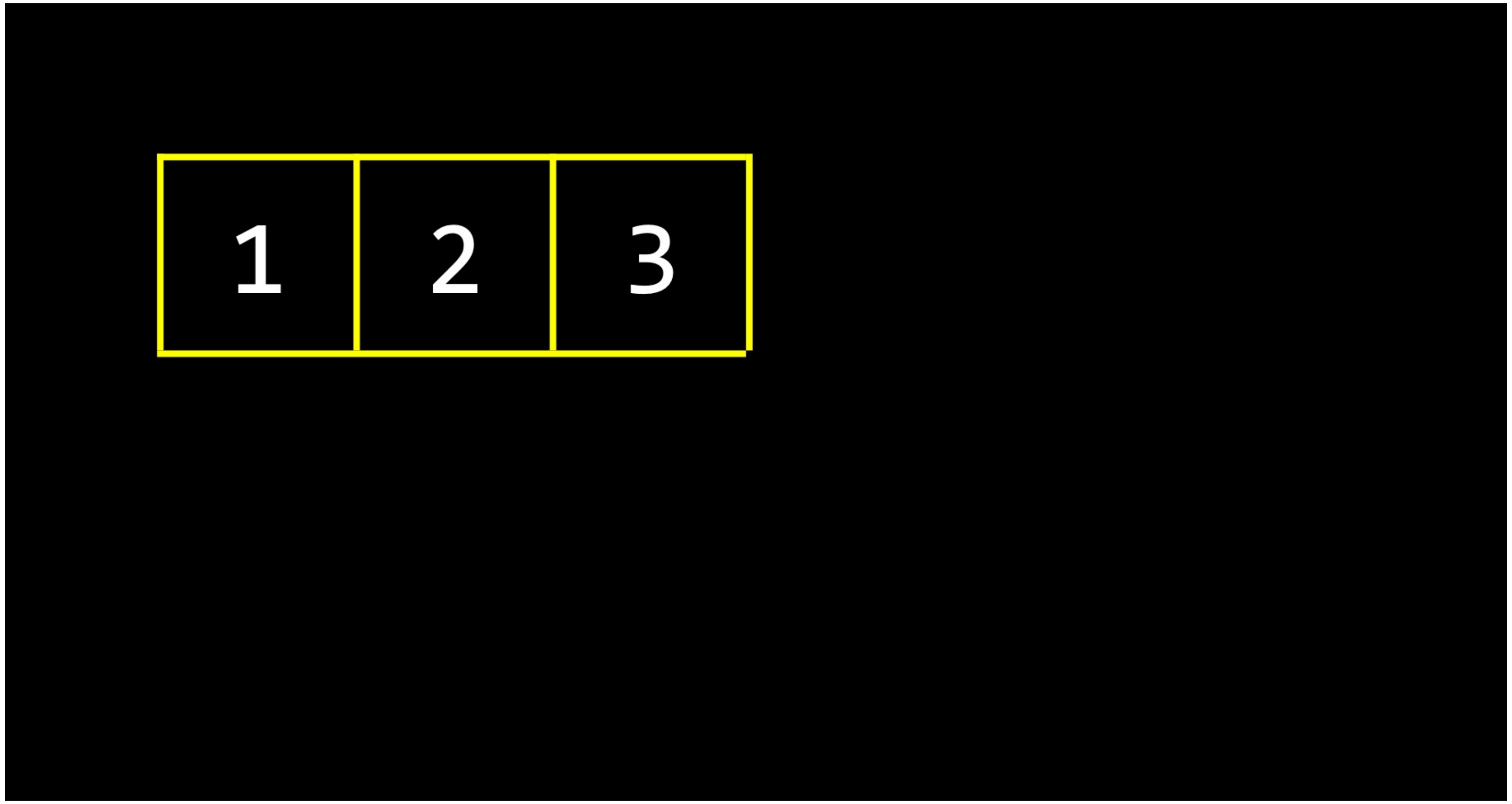
```
#include <iostream>
using namespace std;

int sum(int *arr, int arrayLength)
{
        int sum = 0;
        for(int i = 0; i < arrayLength; i++)
                sum += arr[i]; // Same as *(arr + i)
        return sum;
}

int main()
{
        // Declare and initialize an array of size 5
        int foo[5] = {1, 2, 3, 4, 5};

        cout << sum(foo, 5) << endl;
        return 0;
}
```

```
15
```

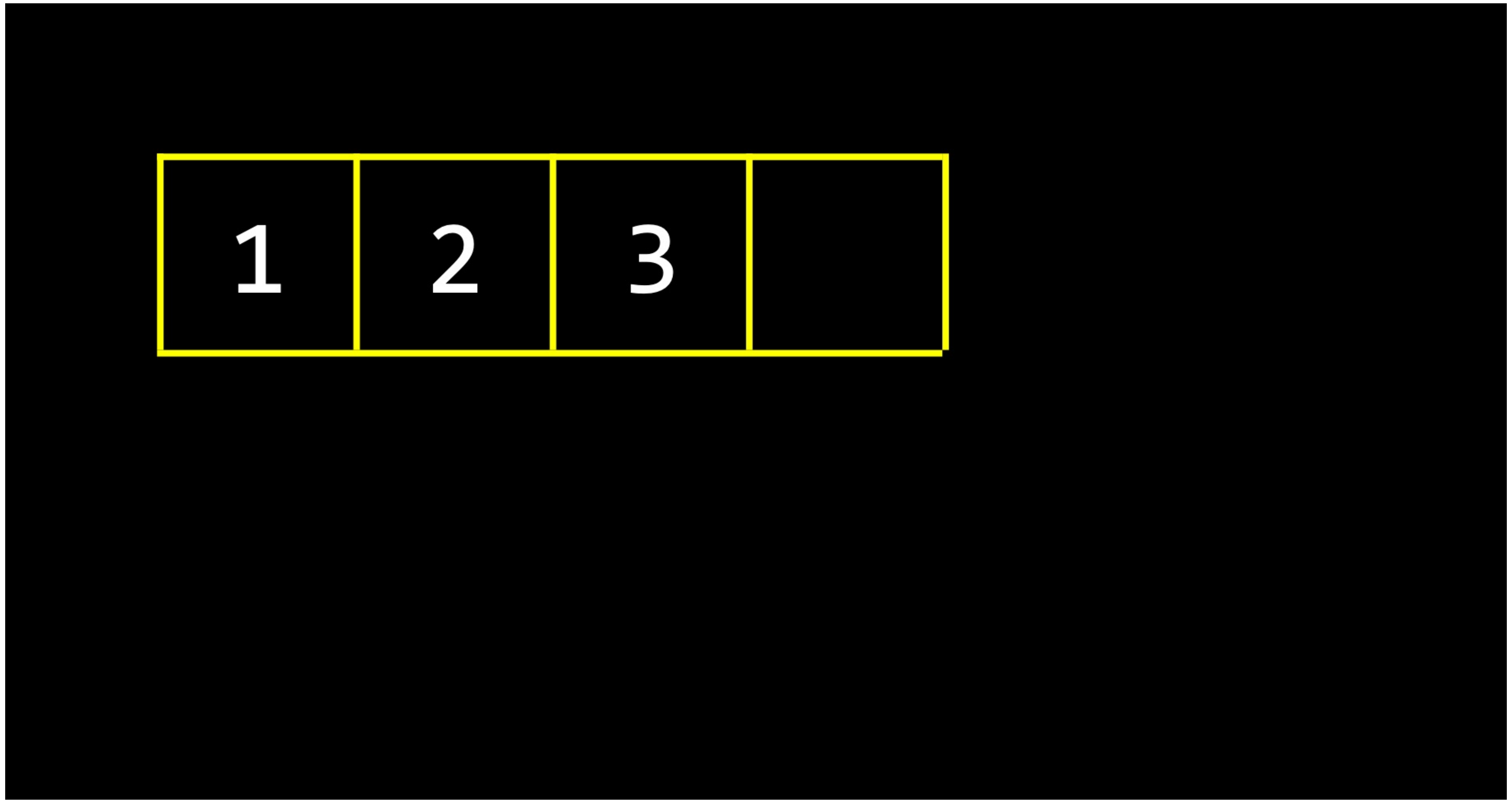
Initialization of an array of size 3



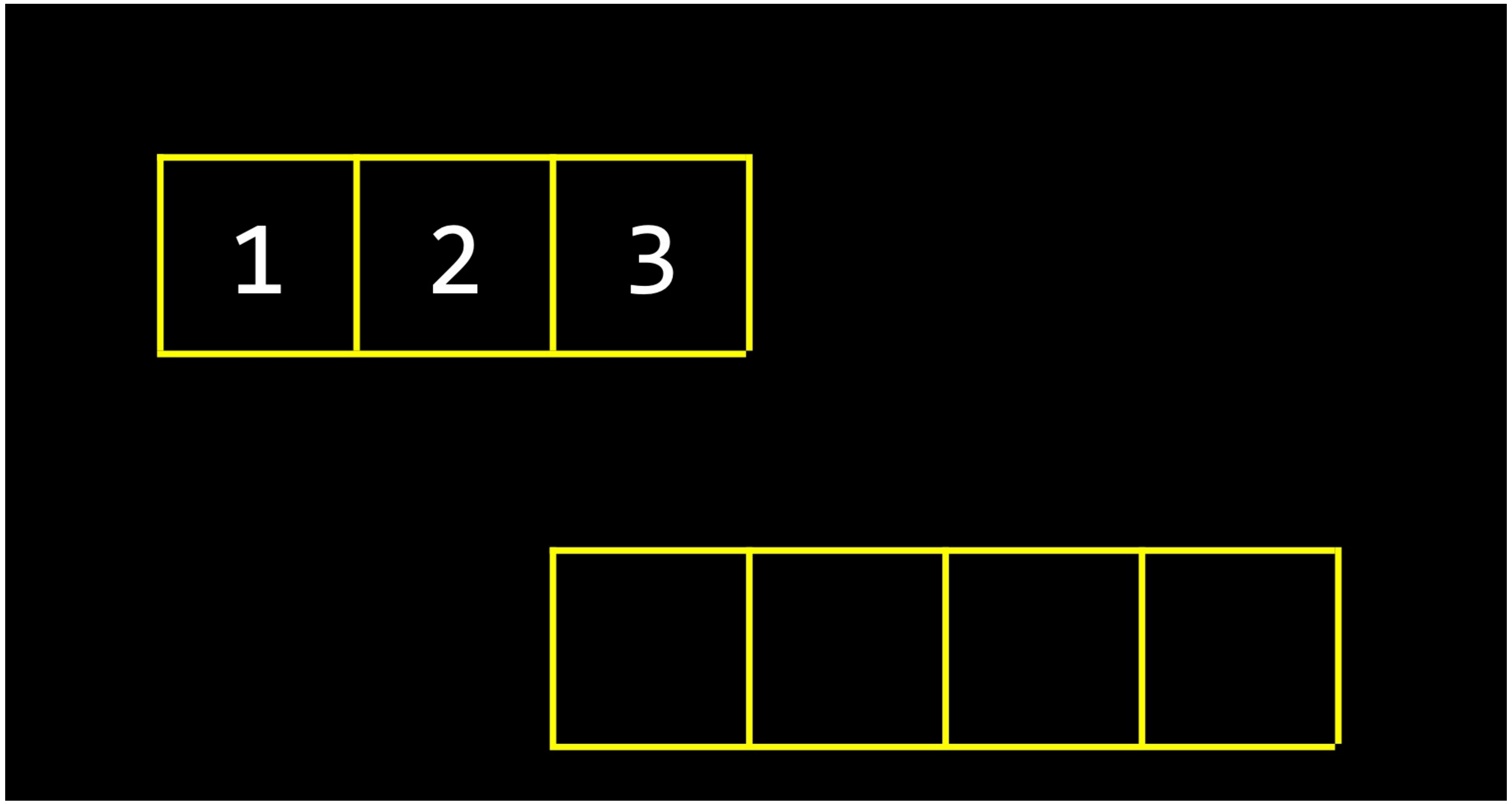
Initialization of an array of size 3



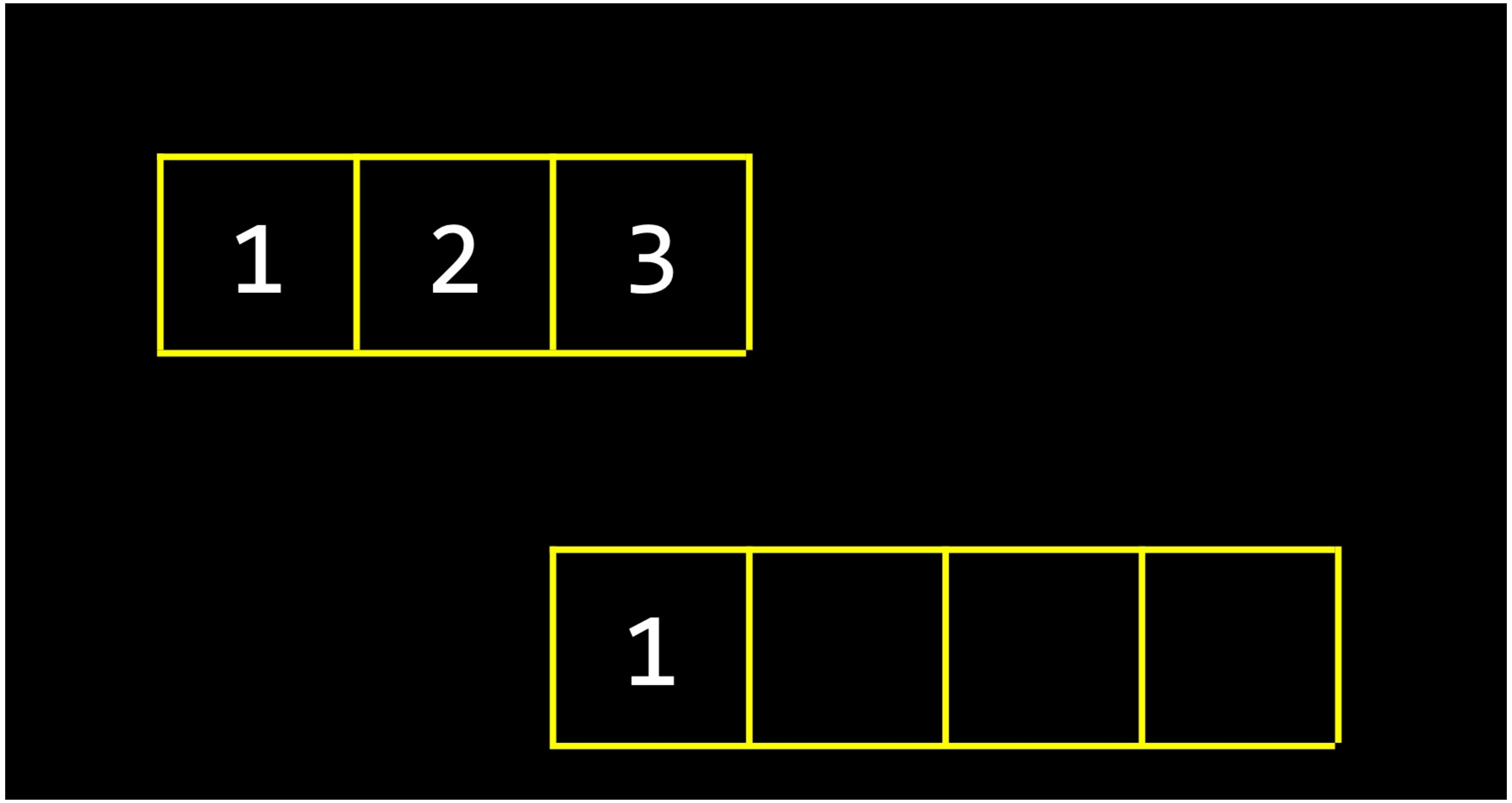
Initialization of an array of size 3 – we cannot simply make the size to 4!



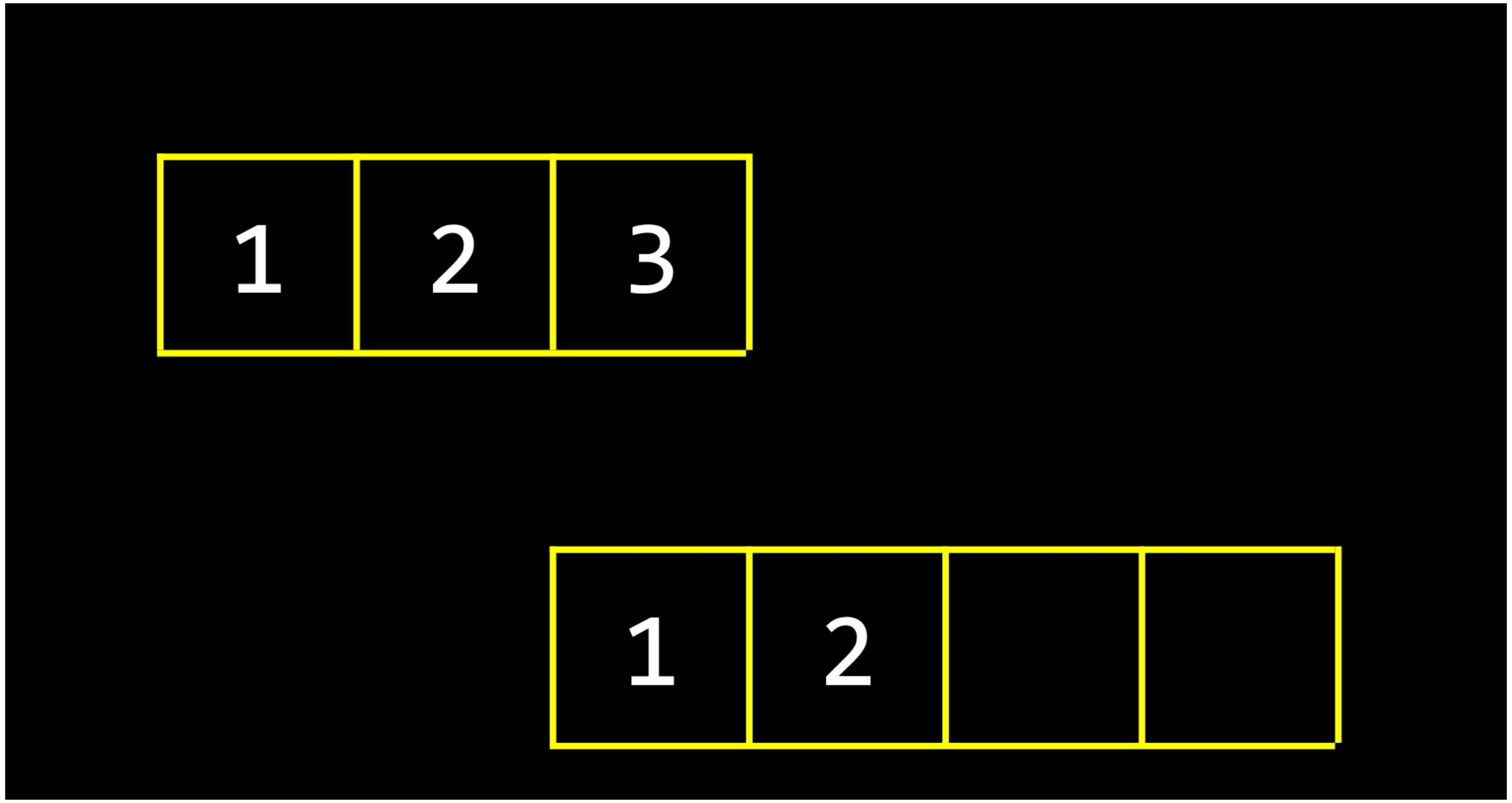
Allocate memory to copy the elements



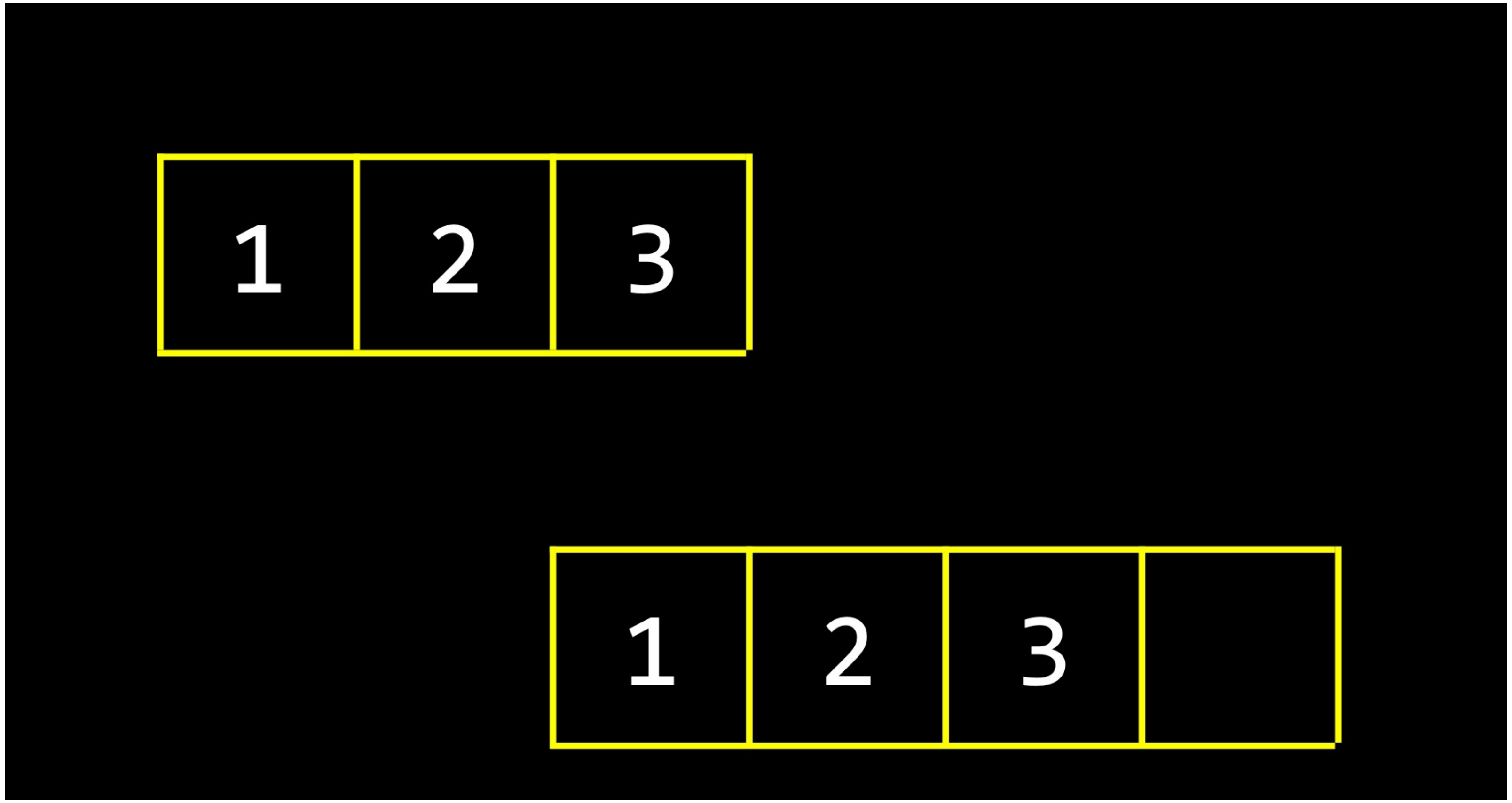
Allocate memory to copy the elements



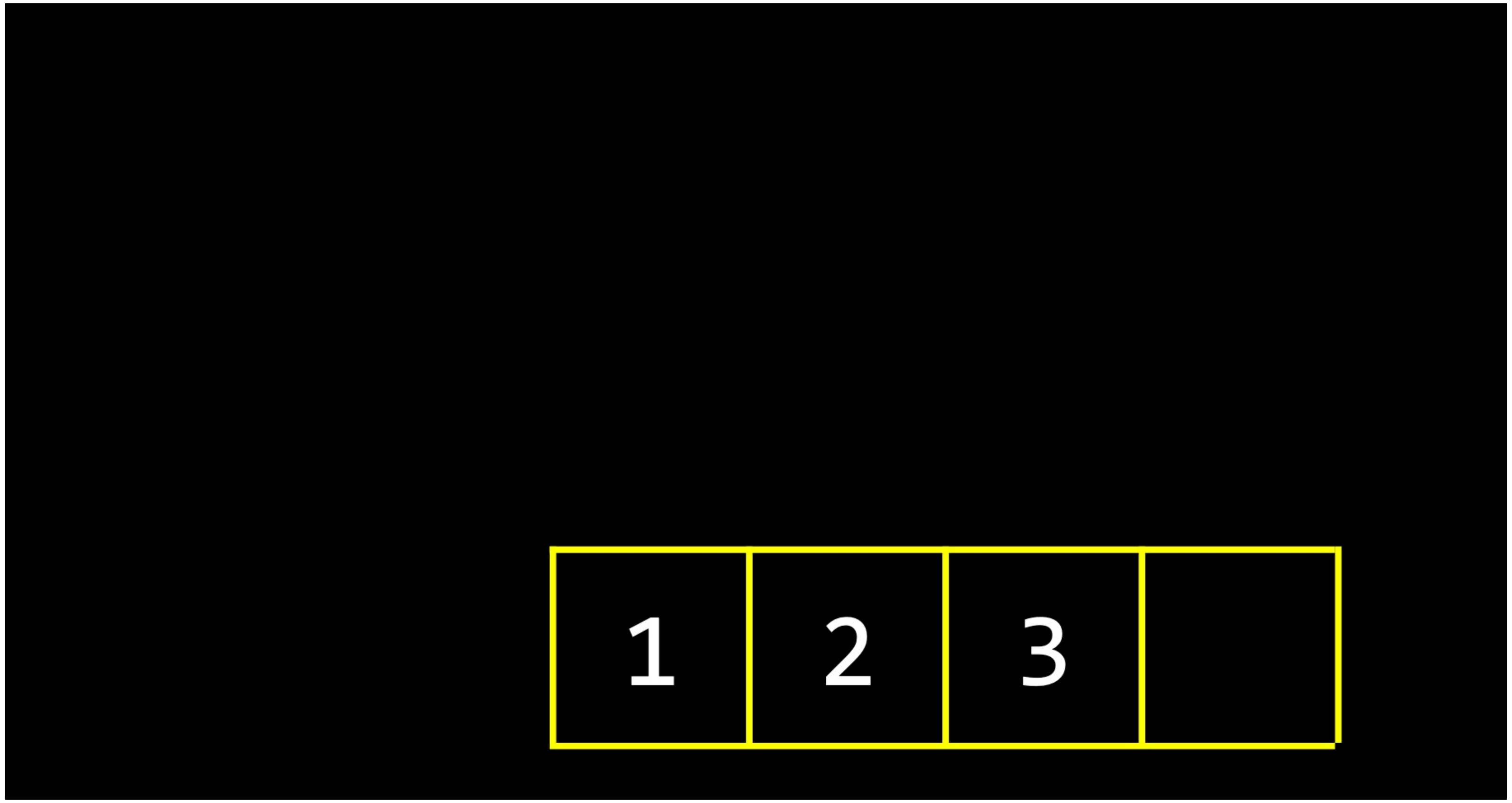
Allocate memory to copy the elements



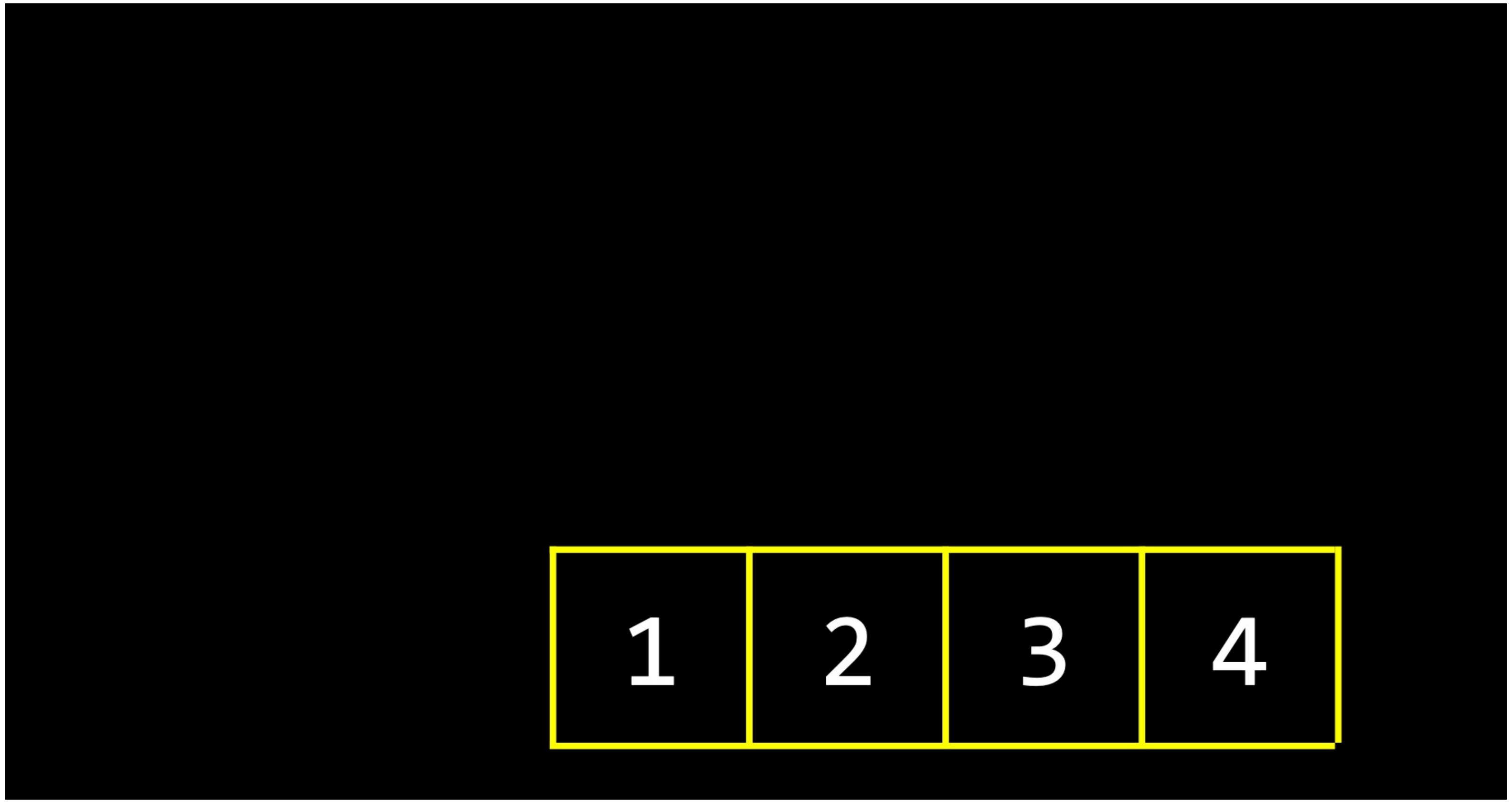
Allocate memory to copy the elements



Allocate memory to copy the elements



Allocate memory to copy the elements



How do we do it?

Introducing

Dynamic Memory Allocation

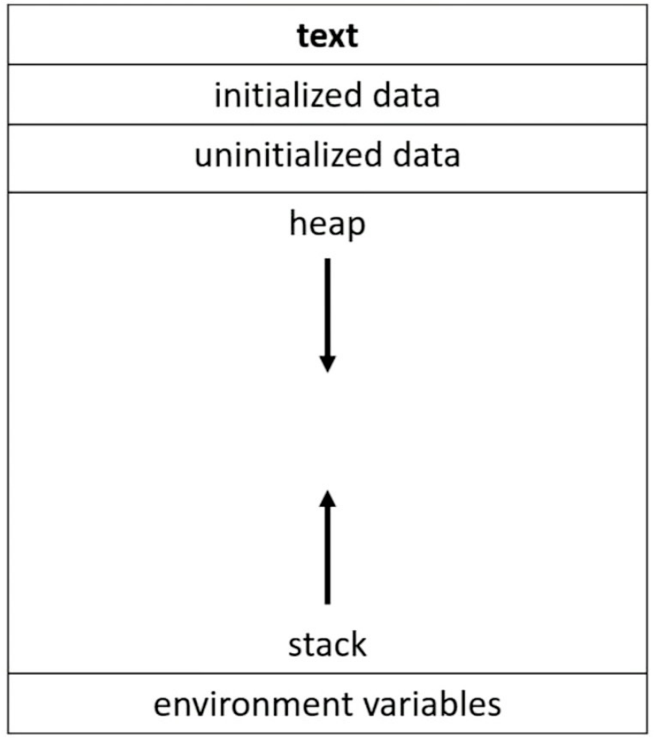
See Code Demonstration!

Dynamic Memory Allocation

- We can use pointers to get access to a block of **dynamically-allocated memory** at *runtime*.
- Dynamic allocated memory comes from a pool of memory known as **heap**.
- Prior to this point, all memory we've been working with has been coming from a pool of memory known as **stack**.

Source: CS50 – Harvard College https://www.youtube.com/watch?v=xa4ugmMDhiE

Dynamic Memory Allocation



Source: CS50 – Harvard College https://www.youtube.com/watch?v=xa4ugmMDhiE

Dynamic Memory Allocation

- We get this dynamically—allocated memory by making a call to the C/C++ standard library fuction `malloc()`, passing as its parameter the number of bytes requested.
- After obtaining memory for you (if it can), `malloc()` will return a pointer to that memory.
- What if `malloc()` cannot give you memory? Then, it will hand you back NULL.

Source: CS50 – Harvard College https://www.youtube.com/watch?v=xa4ugmMDhiE

Dynamic Memory Allocation

- Caution: Dynamically-allocated memory is not automatically returned to the system for later use when the function in which it's created finishes execution.
- When you finish working with dynamically-allocated memory, you must use the C/C++ standard library function `free()` to actually free it.

Source: CS50 – Harvard College https://www.youtube.com/watch?v=xa4ugmMDhiE

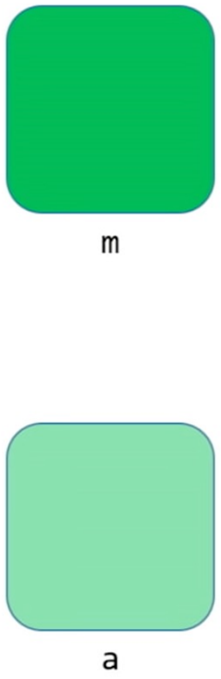
Dynamic Memory Allocation

```
int m;
```

Source: CS50 – Harvard College https://www.youtube.com/watch?v=xa4ugmMDhiE

Dynamic Memory Allocation

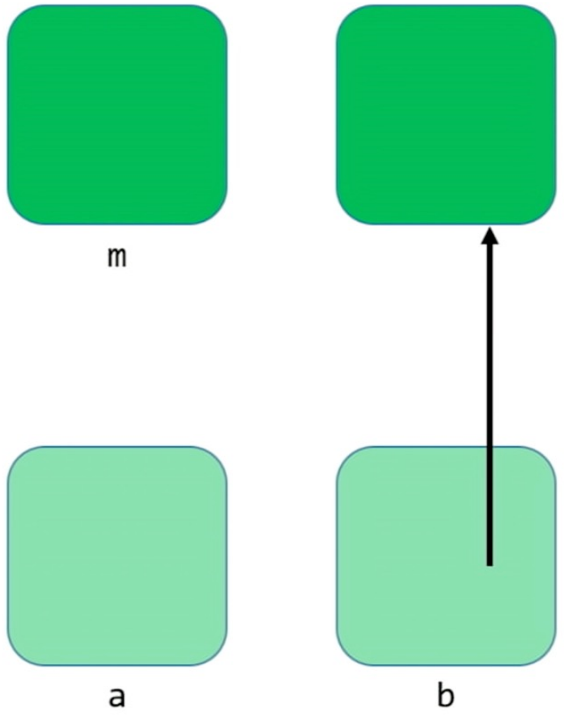
```
int m;
int* a;
```



Source: CS50 – Harvard College https://www.youtube.com/watch?v=xa4ugmMDhiE

Dynamic Memory Allocation

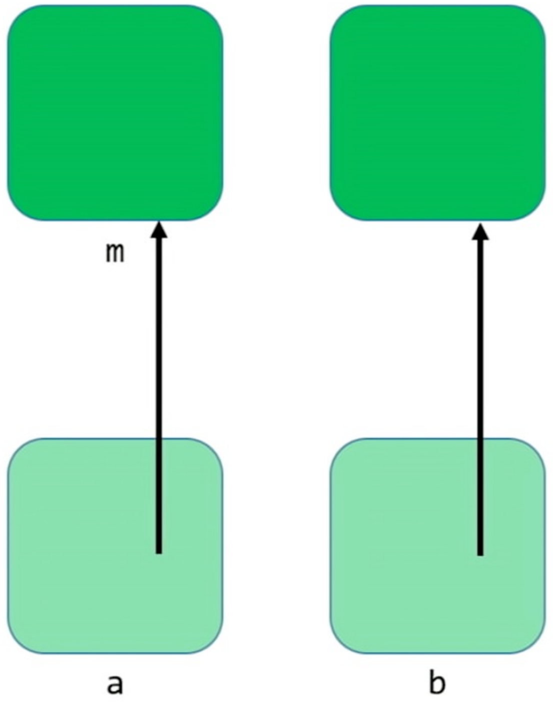
```
int m;
int* a;
int* b = malloc(sizeof(int));
```



Source: CS50 – Harvard College https://www.youtube.com/watch?v=xa4ugmMDhiE

Dynamic Memory Allocation

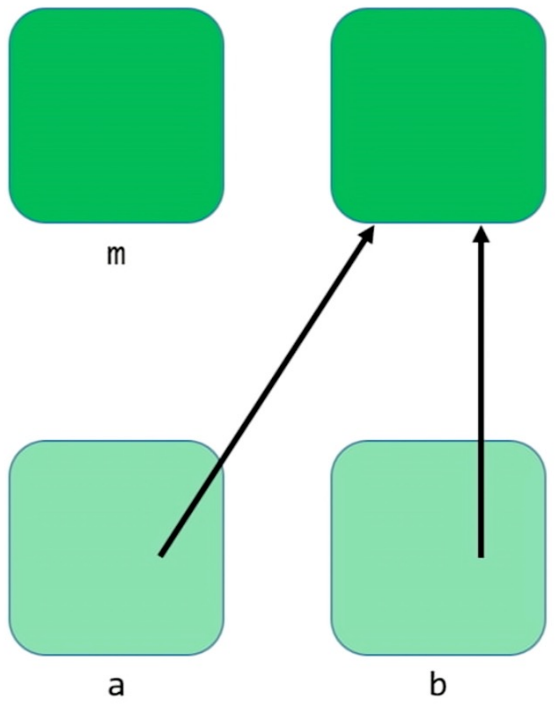
```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
```



Source: CS50 – Harvard College https://www.youtube.com/watch?v=xa4ugmMDhiE

Dynamic Memory Allocation

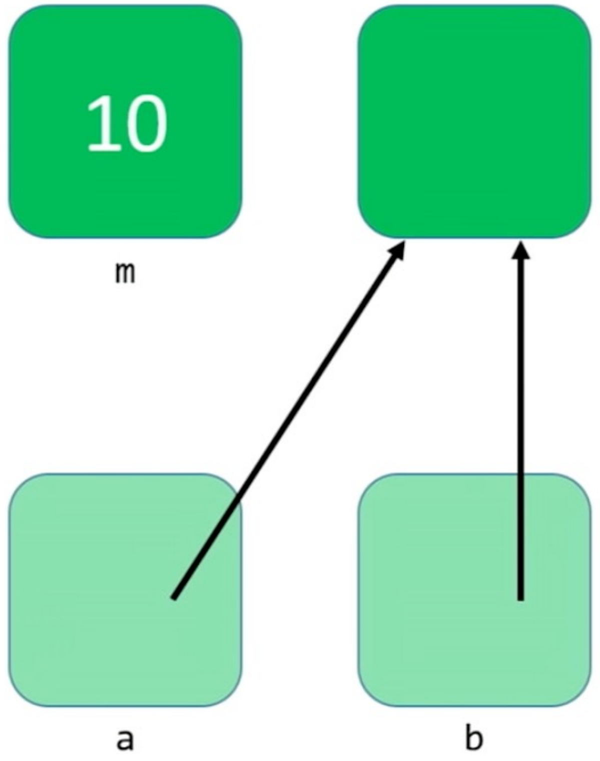
```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
```



Source: CS50 – Harvard College https://www.youtube.com/watch?v=xa4ugmMDhiE

Dynamic Memory Allocation

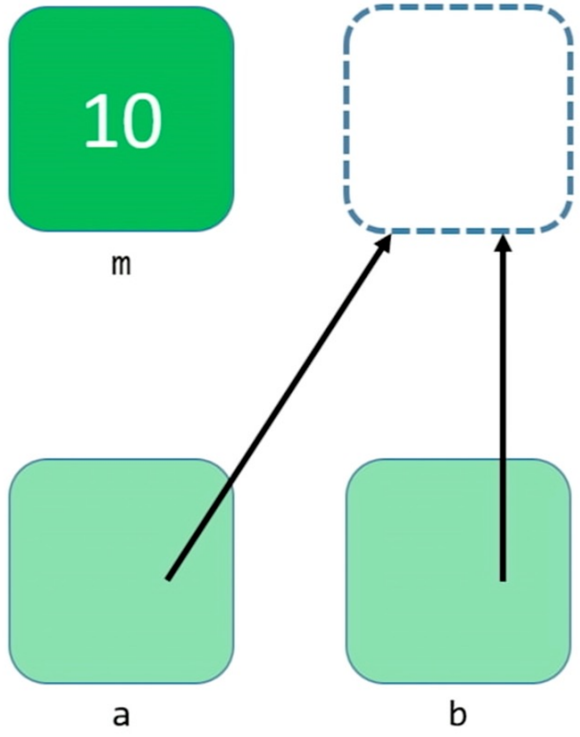
```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
m = 10;
```



Source: CS50 – Harvard College https://www.youtube.com/watch?v=xa4ugmMDhiE

Dynamic Memory Allocation

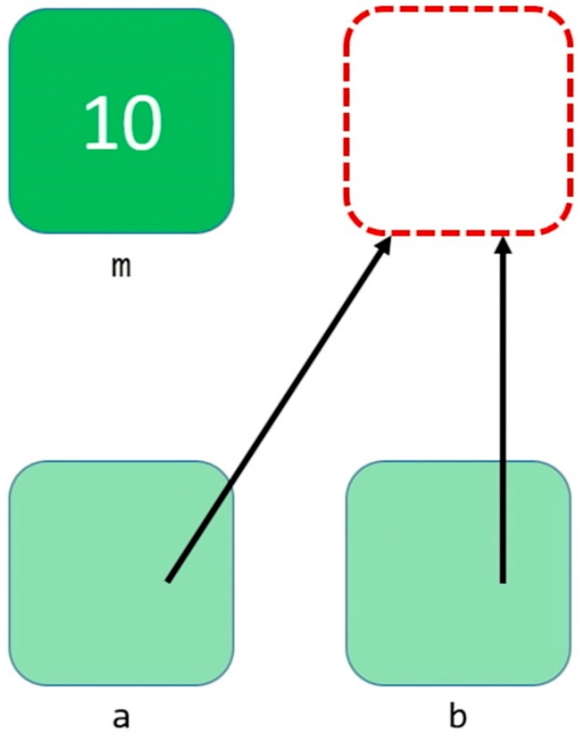
```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
m = 10;
*b = m + 2;
free(b);
```



Source: CS50 – Harvard College https://www.youtube.com/watch?v=xa4ugmMDhiE

Dynamic Memory Allocation

```
int m;
int* a;
int* b = malloc(sizeof(int));
a = &m;
a = b;
m = 10;
*b = m + 2;
free(b);
*a = 11;
```



Source: CS50 – Harvard College https://www.youtube.com/watch?v=xa4ugmMDhiE

Remarks

- Reference
  - MIT 6.096 Introduction to C++
  - This is CS50x, Dr. David J. Malan. https://cs50.harvard.edu/x/2020/