Functions

Ahsan Ayub

Ph.D. Student, Department of Computer Science

Graduate Research Assistant, CEROC

CSC 1300: Introduction to Programming

Friday, October 1, 2021

Simple Problem-Solving Tasks

Write a C++ program that will compute 3^4.

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

int main()
{
        int threeExpFour = 1;
        for (int i = 1; i <= 4; i++)
                threeExpFour *= 3;
        cout << threeExpFour << endl;
        return 0;
}
```

Simple Problem-Solving Tasks

Write a C++ program that will compute 3^4 and 6^5.

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

int main()
{
     int threeExpFour = 1, sixExpFive = 1;
     for (int i = 1; i <= 4; i++)
        threeExpFour *= 3;
     cout << threeExpFour << endl;
     for (int i = 1; i <= 5; i++)
         sixExpFive *= 6;
     cout << sixExpFive << endl;
     return 0;
}
```

Simple Problem-Solving Tasks

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

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

int main()
{
     int threeExpFour = 1, sixExpFive = 1, twelveExpTen = 1;
     for (int i = 1; i <= 4; i++)
        threeExpFour *= 3;
     cout << threeExpFour << endl;
     for (int i = 1; i <= 5; i++)
         sixExpFive *= 6;
     cout << sixExpFive << endl;
     for (int i = 1; i <= 10; i++)
         twelveExpTen *= 12;
     cout << twelveExpTen << endl;
     return 0;
}
```

Simple Problem-Solving Tasks

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Repeating the same functionality again and again!

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

int main()
{
     int threeExpFour = 1, sixExpFive = 1, twelveExpTen = 1;
     for (int i = 1; i <= 4; i++)
        threeExpFour *= 3;
     cout << threeExpFour << endl;
     for (int i = 1; i <= 5; i++)
         sixExpFive *= 6;
     cout << sixExpFive << endl;
     for (int i = 1; i <= 10; i++)
         twelveExpTen *= 12;
     cout << twelveExpTen << endl;
     return 0;
}
```

Introducing Functions

- A block of code that performs a specific task.
- Using functions, we divide a complex problem into smaller chunks to makes our program easy to understand and reusable.
- There are to types of functions:
  - Standard Library Function – Predefined in C++
  - User-defined Functions – we define it

C++ Function Declaration

```
returnType FunctionName (arguement1, arguement2, …)
{
        // Function body
}
```

C++ Function Declaration

Function Signature

```
returnType FunctionName (arguement1, arguement2, …)
{
        // Function body
}
```

C++ Function Declaration

short, int, long long

float, double

void

```
returnType FunctionName (arguement1, arguement2, …)
{
        // Function body
}
```

Using Function

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     int sixExpFive = RaiseToPower(6,5);
     cout << sixExpFive << endl;
     int twelveExpTen = RaiseToPower(12,10);
     cout << twelveExpTen << endl;
     return 0;
}
```

Using Function

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

Using Function

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

Using Function

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

Using Function

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

function invocation

Using Function

result	base	exponent
1	3	4

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
    int result = 1;
    for(int i = 1; i <= exponent; i++)
        result *= base;
    return result;
}

int main()
{
    int threeExpFour = RaiseToPower(3,4);
    cout << threeExpFour << endl;
    return 0;
}
```

Using Function

result	base	exponent
1	3	4

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

Using Function

result	base	exponent
~~1~~ 3	3	4

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
     result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

Using Function

result	base	exponent
3	3	4

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

Using Function

result	base	exponent
~~3~~ 9	3	4

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
     result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

Using Function

result	base	exponent
9	3	4

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

Using Function

result	base	exponent
~~9~~ 27	3	4

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
     result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

Using Function

result	base	exponent
27	3	4

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

Using Function

result	base	exponent
~~27~~ 81	3	4

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
     result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

Using Function

result	base	exponent
81	3	4

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

Using Function

result	base	exponent
81	3	4

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}
```

return statement

```
int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

threeExpFour
81

Using Function

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

threeExpFour
81

Using Function

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

threeExpFour
81

Using Function

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Let's walkthrough the program!

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     return 0;
}
```

Using Function

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

Thus, we can continue reusing the function however long we want.

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     int sixExpFive = RaiseToPower(6,5);
     cout << sixExpFive << endl;
     int twelveExpTen = RaiseToPower(12,10);
     cout << twelveExpTen << endl;
     return 0;
}
```

Using Function

Write a C++ program that will compute 3^4, 6^5, and 12^{10}.

The code could be rewritten with declaring the function signature at the top (before the main function) and defining the function at the bottom (after the main function).

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

int RaiseToPower(int, int); // Declaration or Prototype

int main()
{
     int threeExpFour = RaiseToPower(3,4);
     cout << threeExpFour << endl;
     int sixExpFive = RaiseToPower(6,5);
     cout << sixExpFive << endl;
     int twelveExpTen = RaiseToPower(12,10);
     cout << twelveExpTen << endl;
     return 0;
}

int RaiseToPower(int base, int exponent) // Definition
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}
```

Output Tracing

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     cout << RaiseToPower(2,5) << endl;
     return 0;
}
```

```
??
```

Output Tracing

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     cout << RaiseToPower(2,5) << endl;
     return 0;
}
```

```
32
```

Output Tracing

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     cout << RaiseToPower(5,2) << endl;
     return 0;
}
```

```
??
```

Output Tracing

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

int RaiseToPower(int base, int exponent)
{
     int result = 1;
     for(int i = 1; i <= exponent; i++)
        result *= base;
     return result;
}

int main()
{
     cout << RaiseToPower(5,2) << endl;
     return 0;
}
```

```
25
```

Output Tracing

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

double SumOfTwoNumbers(double a, double b)
{
        return (a + b);
}

int main()
{
        cout << SumOfTwoNumbers(10.5,50.25) << endl;
        cout << SumOfTwoNumbers(7.75,8.125) << endl;
        return 0;
}
```

```
??
```

Output Tracing

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

double SumOfTwoNumbers(double a, double b)
{
        return (a + b);
}

int main()
{
        cout << SumOfTwoNumbers(10.5,50.25) << endl;
        cout << SumOfTwoNumbers(7.75,8.125) << endl;
        return 0;
}
```

```
60.75
15.875
```

Output Tracing

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

bool IsOdd(int number)
{
        if(number % 2 != 0)
                return true;
        else
                return false;
}

int main()
{
        if(IsOdd(10))
                cout << “10 is an odd number.\n”;
        else
                cout << “10 is an even number.\n”;
        return 0;
}
```

```
??
```

Output Tracing

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

bool IsOdd(int number)
{
        if(number % 2 != 0)
                return true;
        else
                return false;
}

int main()
{
        if(IsOdd(10))
                cout << “10 is an odd number.\n”;
        else
                cout << “10 is an even number.\n”;
        return 0;
}
```

```
10 is an even number.
```

Output Tracing

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

void CheckEvenOrOdd(int number)
{
        if(number % 2 != 0)
                cout << number << “ is an odd number.” << endl;
        else
                cout << number << “ is an even number.” << endl;;
}

int main()
{
        CheckEvenOrOdd(10);
        CheckEvenOrOdd(15);
        return 0;
}
```

```
??
```

Note: If no values are returned, we give the function a **void** return type.

Output Tracing

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

void CheckEvenOrOdd(int number)
{
        if(number % 2 != 0)
                cout << number << “ is an odd number.” << endl;
        else
                cout << number << “ is an even number.” << endl;;
}

int main()
{
        CheckEvenOrOdd(10);
        CheckEvenOrOdd(15);
        return 0;
}
```

```
10 is an even number.
15 is an odd number.
```

Output Tracing

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

void MyFunction(int a, double b)
{
        cout << a << “ ” << b << endl;
}

int main()
{
        MyFunction(10, 10.5);
        MyFunction(10.5, 10);
        return 0;
}
```

```
??
```

Output Tracing

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

void MyFunction(int a, double b)
{
        cout << a << “ ” << b << endl;
}

int main()
{
        MyFunction(10, 10.5);
        MyFunction(10.5, 10);
        return 0;
}
```

```
10 10.5
10 10
```

Note: Argument type matters!

Output Tracing

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

void PrintNumberIfEven(int number)
{
        if(number % 2 != 0)
                return;
        cout << number << “ is an even number.\n”;
}

int main()
{
        PrintNumberIfEven(111);
        PrintNumberIfEven(100);
        return 0;
}
```

```
??
```

Output Tracing

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

void PrintNumberIfEven(int number)
{
        if(number % 2 != 0)
                return;
        cout << number << “ is an even number.\n”;
}

int main()
{
        PrintNumberIfEven(111);
        PrintNumberIfEven(100);
        return 0;
}
```

```
100 is an even number.
```

Note: Return statements don’t necessarily need to be at the end. Function returns as soon as a return statement is executed.

Output Tracing

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

int Square(int x)
{
        return x*x;
}

int Cube(int x)
{
        return x*Square(x);
}

int main()
{
        cout << Cube(2) << endl;
        return 0;
}
```

```
??
```

Output Tracing

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

int Square(int x)
{
        return x*x;
}

int Cube(int x)
{
        return x*Square(x);
}

int main()
{
        cout << Cube(2) << endl;
        return 0;
}
```

```
8
```

Output Tracing

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

int Cube(int x)
{
        return x*Square(x);
}

int Square(int x)
{
        return x*x;
}

int main()
{
        cout << Cube(2) << endl;
        return 0;
}
```

```
??
```

Output Tracing

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

int Cube(int x)
{
        return x*Square(x);
}

int Square(int x)
{
        return x*x;
}

int main()
{
        cout << Cube(2) << endl;
        return 0;
}
```

```
Error
```

Note: Function declarations need to occur before invocations. Solution: Reorder the function declaration or use prototype to inform the compile that it'll be implemented later on.

Output Tracing

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

int Square(int);   // Prototype or declaration

int Cube(int x)
{
        return x*Square(x);
}

int Square(int x)
{
        return x*x;
}

int main()
{
        cout << Cube(2) << endl;
        return 0;
}
```

```
8
```

Output Tracing

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

// Declared as the global variable – accessible from all over the code
int NumberOfCalls = 0;
void MyFunction()
{
        NumberOfCalls += 1; // Accessing the global variable
}

int main()
{
        MyFunction();
        MyFunction();
        MyFunction();
        cout << NumberOfCalls << endl;
        return 0;
}
```

```
??
```

Output Tracing

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

// Declared as the global variable – accessible from all over the code
int NumberOfCalls = 0;
void MyFunction()
{
        NumberOfCalls += 1; // Accessing the global variable
}

int main()
{
        MyFunction();
        MyFunction();
        MyFunction();
        cout << NumberOfCalls << endl;
        return 0;
}
```

```
3
```

Output Tracing

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

// Declared as the global variable – accessible from all over the code
int a = 20;
int Sum(int a, int b)       // Both a and b are local variables
{
        return (a + b);
}

int main()
{
        cout << Sum(10,20) << endl;
        cout << (a + 20) << endl;
        return 0;
}
```

```
??
```

Output Tracing

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

// Declared as the global variable – accessible from all over the code
int a = 20;
int Sum(int a, int b)       // Both a and b are local variables
{
        return (a + b);
}

int main()
{
        cout << Sum(10,20) << endl;
        cout << (a + 20) << endl;
        return 0;
}
```

```
30
40
```

Note: Functions arguments are treated as local variables with-in a function and they take precedence over the global variables.

The Benefits of Defining Functions

- Readability: `sqrt(5)` is clearer than copy-pasting in an algorithm to compute the square root.
- Maintainability: To change the functionalities, we just need to change the function as opposed to modify the code everywhere we used.
- Code Reuse: We can use the algorithms that others have implemented.

Function Overloading

- Two or more functions with the same name but different arguments.
- The function called is the one whose arguments match the invocation

```
void display(int var1, double var2) {
    // code
}

void display(double var) {
    // code
}

void display(int var) {
    // code
}

int main() {
    int a = 5;
    double b = 5.5;

    display(a);

    display(b);

    display(a, b);

    ... ...

}
```

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

Output Tracing

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

void UserInformation(int age)
{
        cout << “My age is ” << age << endl;
}

void UserInformation(int age, int year)
{
        cout << “My age is ” << age << “ and I was born in ” << year << endl;
}

int main()
{
        UserInformation(82);
        UserInformation(82,1930);
        return 0;
}
```

```
??
```

Output Tracing

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

void UserInformation(int age)
{
        cout << “My age is ” << age << endl;
}

void UserInformation(int age, int year)
{
        cout << “My age is ” << age << “ and I was born in ” << year << endl;
}

int main()
{
        UserInformation(82);
        UserInformation(82,1930);
        return 0;
}
```

```
My age is 82
My age is 82 and I was born in 1930.
```

Output Tracing

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

double Absolute(double x)
{
            if(x < 0.0)
                        return ((-1)*x);
            return x;
}

int Absolute(int x)
{
            if(x < 0)
                        return ((-1)*x);
            return x;
}

int main()
{
            cout << Absolute(82) << endl;
            cout << Absolute(-82.558) << endl;
            return 0;
}
```

```
??
```

Output Tracing

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

double Absolute(double x)
{
            if(x < 0.0)
                        return ((-1)*x);
            return x;
}

int Absolute(int x)
{
            if(x < 0)
                        return ((-1)*x);
            return x;
}

int main()
{
            cout << Absolute(82) << endl;
            cout << Absolute(-82.558) << endl;
            return 0;
}
```

```
82
82.558
```

Pass by Value

- So far, we have been passing everything by value
  - Make a copy of the variable
  - Modify the variable within the function body
  - This process doesn't automatically change the variable's value from where the function has been invoked.

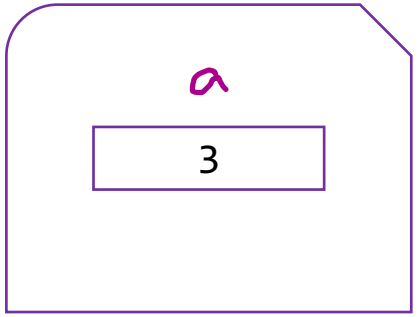
```
#include <iostream>
using namespace std;

void Increment(int x)
{
        x += 1;
        cout << "x in Increment: " << x << endl;
}

int main()
{
        int a = 3;
        Increment(a);
        cout << "a in Main: " << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 3
```

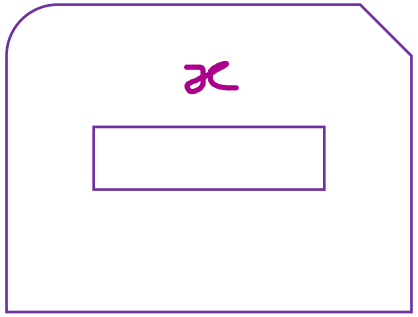
Pass by Value

- So far, we have been passing everything by value
  - This process doesn't automatically change the variable's value from where the function has been invoked.

Main function scope



Increment function scope



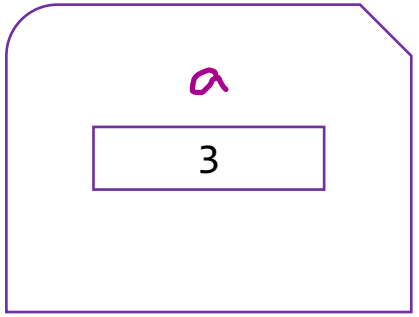

```
#include <iostream>
using namespace std;

void Increment(int x)
{
        x += 1;
        cout << "x in Increment: " << x << endl;
}

int main()
{
        int a = 3;
        Increment(a);
        cout << "a in Main: " << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 3
```

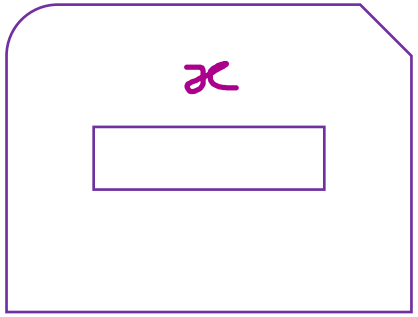
Pass by Value

- So far, we have been passing everything by value
  - This process doesn't automatically change the variable's value from where the function has been invoked.

Main function scope



Increment function scope



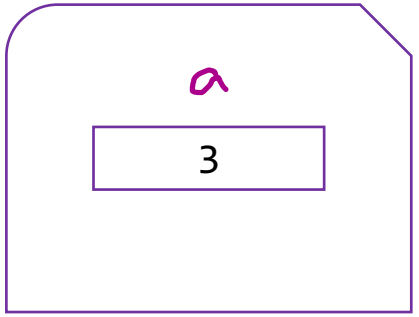

```
#include <iostream>
using namespace std;

void Increment(int x)
{
        x += 1;
        cout << "x in Increment: " << x << endl;
}

int main()
{
        int a = 3;
        Increment(a);
        cout << "a in Main: " << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 3
```

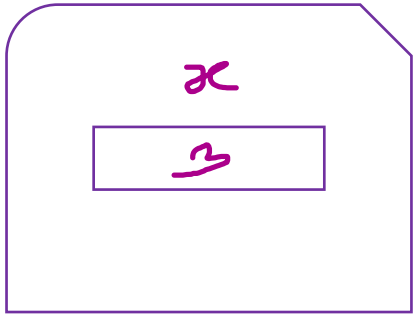
Pass by Value

- So far, we have been passing everything by value
  - This process doesn't automatically change the variable's value from where the function has been invoked.

Main function scope



Increment function scope



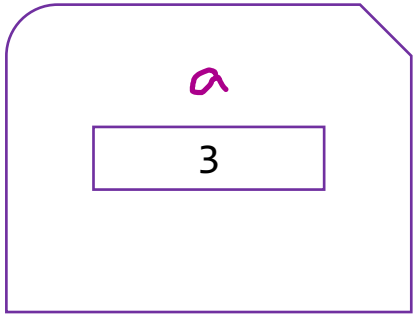

```
#include <iostream>
using namespace std;

void Increment(int x)
{
        x += 1;
        cout << "x in Increment: " << x << endl;
}

int main()
{
        int a = 3;
        Increment(a);
        cout << "a in Main: " << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 3
```

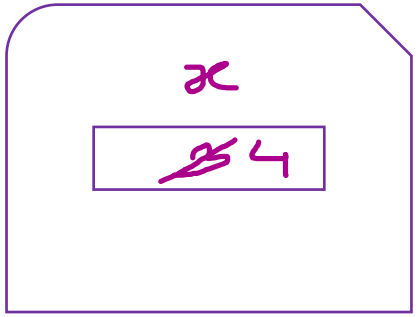
Pass by Value

- So far, we have been passing everything by value
  - This process doesn't automatically change the variable's value from where the function has been invoked.

Main function scope



Increment function scope



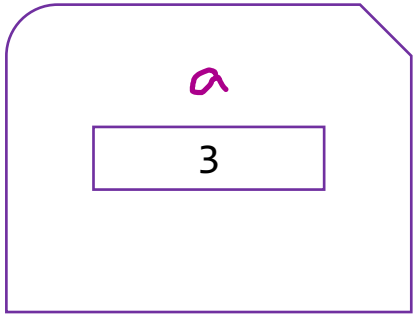

```
#include <iostream>
using namespace std;

void Increment(int x)
{
        x += 1;
        cout << "x in Increment: " << x << endl;
}

int main()
{
        int a = 3;
        Increment(a);
        cout << "a in Main: " << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 3
```

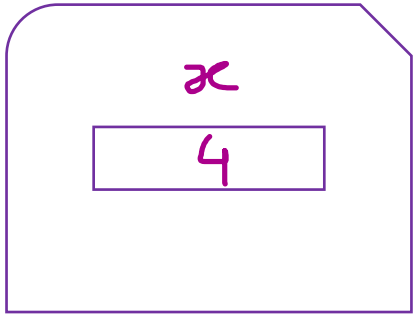
Pass by Value

- So far, we have been passing everything by value
  - This process doesn't automatically change the variable's value from where the function has been invoked.

Main function scope



Increment function scope




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

void Increment(int x)
{
        x += 1;
        cout << "x in Increment: " << x << endl;
}

int main()
{
        int a = 3;
        Increment(a);
        cout << "a in Main: " << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 3
```

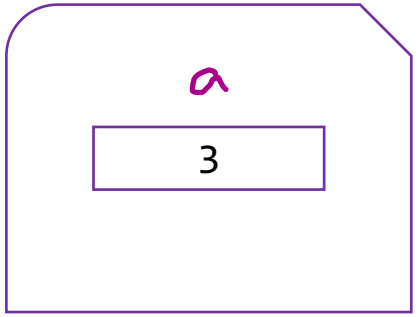
Pass by Value

- So far, we have been passing everything by value
  - This process doesn't automatically change the variable's value from where the function has been invoked.

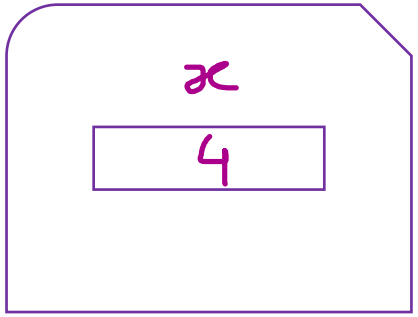
```
#include <iostream>
using namespace std;

void Increment(int x)
{
        x += 1;
        cout << "x in Increment: " << x << endl;
}

int main()
{
        int a = 3;
        Increment(a);
        cout << "a in Main: " << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 3
```

Main function scope

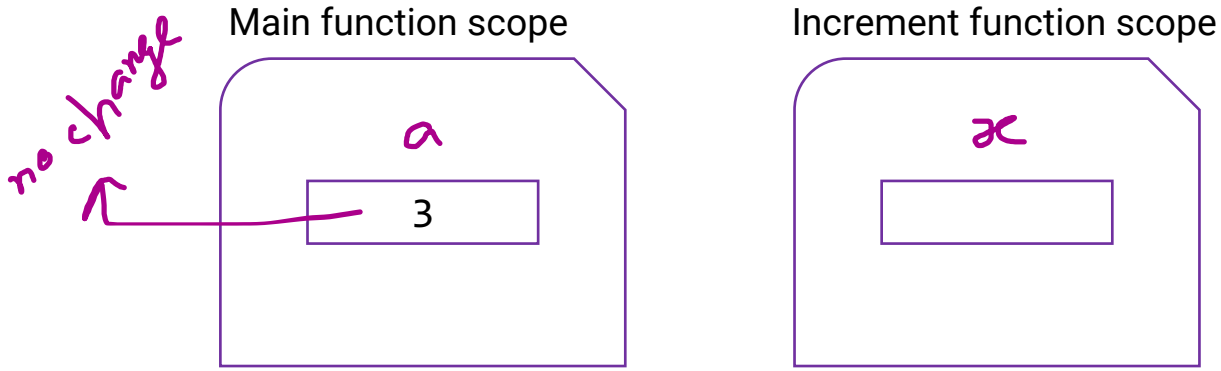


Increment function scope



Pass by Value

- So far, we have been passing everything by value
  - This process doesn't automatically change the variable's value from where the function has been invoked.



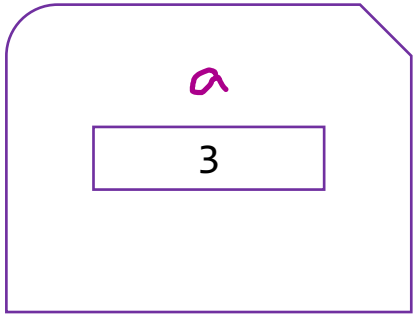

```
#include <iostream>
using namespace std;

void Increment(int x)
{
        x += 1;
        cout << "x in Increment: " << x << endl;
}

int main()
{
        int a = 3;
        Increment(a);
        cout << "a in Main: " << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 3
```

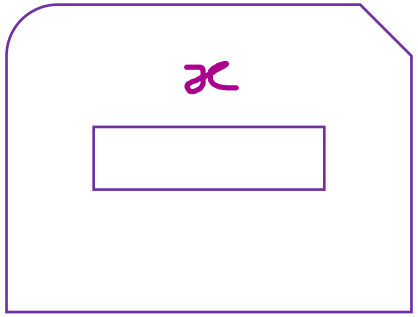
Pass by Value

- So far, we have been passing everything by value
  - This process doesn't automatically change the variable's value from where the function has been invoked.

Main function scope



Increment function scope



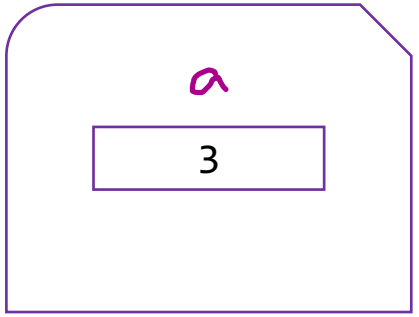

```
#include <iostream>
using namespace std;

void Increment(int x)
{
        x += 1;
        cout << "x in Increment: " << x << endl;
}

int main()
{
        int a = 3;
        Increment(a);
        cout << "a in Main: " << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 3
```

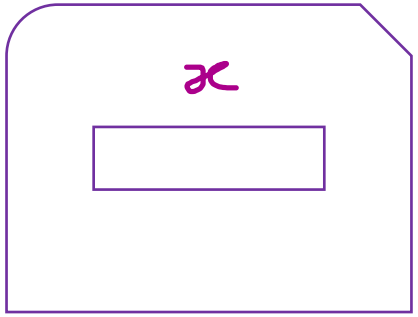
Pass by Value

- So far, we have been passing everything by value
  - This process doesn't automatically change the variable's value from where the function has been invoked.

Main function scope



Increment function scope




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

void Increment(int x)
{
        x += 1;
        cout << "x in Increment: " << x << endl;
}

int main()
{
        int a = 3;
        Increment(a);
        cout << "a in Main: " << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 3
```

Pass by Value

- So far, we have been passing everything by value
  - Make a copy of the variable
  - Modify the variable within the function body
  - This process doesn't automatically change the variable's value from where the function has been invoked.

Note: With the assignment operation and a return statement, we could get it to work as per the example on right.

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

int Increment(int x)
{
        x += 1;
        cout << "x in Increment: " << x << endl;
        return x;
}

int main()
{
        int a = 3;
        a = Increment(a);
        cout << "a in Main: " << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 4
```

Limitation of a `return` statement

- In C/C++, the return statement only allows us to return 1 value.
- Passing output variables by reference overcomes this limitation.

Recap: Capacity of Datatypes in C/C++

- bool – 1 byte | 00000001 | bool var = true
- char – 1 byte | 01000001 | char var = 'A'
- float – 4 bytes | <8 bits> | <8 bits> | <8 bits> | <8 bits>
- double – 8 bytes | <8 bits> | <8 bits> | <8 bits> | <8 bits> | ..... | 8th <8 bits>
- int – 4 bytes | <8 bits> | <8 bits> | <8 bits> | <8 bits>
- long long – 8 bytes | <8 bits> | <8 bits> | <8 bits> | <8 bits> | ..... | 8th <8 bits>

Memory Map Representation

```
int myIntegerVariable = 100;
```

myIntegerVariable → name

0000000	0000000	0000000	01100100

→ value

140734763609956 → address

Binary Numbers

```
0 1
```

Decimal Numbers

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

Hexadecimal Numbers (base-16 Number System)

0 1 2 3 4 5 6 7 8 9 A B C D E F

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	0

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	1

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	2

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	3

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	4

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	5

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	6

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	7

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	8

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	9

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	A

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	B

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	C

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	D

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
0	F

Hexadecimal Numbers (base-16 Number System)

16^1	16^0
1	0

Hexadecimal Numbers (base-16 Number System)

16	1
F	F

Hexadecimal Numbers (base-16 Number System)

```
16  1
F   F
16*15 + 1*15
```

Hexadecimal Numbers (base-16 Number System)

16	1
F	F
255	

Hexadecimal Numbers (base-16 Number System)

16	1
F	F

$(255)_{10}$

Hexadecimal Numbers (base-16 Number System)

16 1

F F

$(255)_{10}$

$(1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)_2$

Hexadecimal Numbers (base-16 Number System)

1 1 1 1	1 1 1 1
F	F

Hexadecimal and Binary Numbers System

0 0 0 0 0 0 0 0

0 0

Hexadecimal and Binary Numbers System

```
0 0 0 0      0 0 0 1
  0            1
```

Hexadecimal and Binary Numbers System

```
0 0 0 0      0 0 1 0
  0            2
```

Hexadecimal and Binary Numbers System

```
0 0 0 0     0 0 1 1
  0           3
```

Hexadecimal and Binary Numbers System

```
0 0 0 0     0 1 0 0
  0           4
```

Hexadecimal and Binary Numbers System

```
0 0 0 0     0 1 0 1
  0           5
```

Hexadecimal and Binary Numbers System

0 0 0 0	0 1 1 0
0	6

Hexadecimal and Binary Numbers System

```
0 0 0 0     0 1 1 1
  0           7
```

Hexadecimal and Binary Numbers System

0 0 0 0	1 0 0 0
0	8

Hexadecimal and Binary Numbers System

```
0 0 0 0     1 0 0 1
  0           9
```

Hexadecimal and Binary Numbers System

0 0 0 0	1 0 1 0
0	A

Hexadecimal and Binary Numbers System

```
0 0 0 0     1 0 1 1
  0           B
```

Hexadecimal and Binary Numbers System

```
0 0 0 0     1 1 0 0
  0           C
```

Hexadecimal and Binary Numbers System

```
0 0 0 0     1 1 0 1
  0           D
```

Hexadecimal and Binary Numbers System

0 0 0 0	1 1 1 0
0	E

Hexadecimal and Binary Numbers System

```
0 0 0 0     1 1 1 1
  0           F
```

Hexadecimal and Binary Numbers System

```
0 0 0 1     0 0 0 0
  1           0
```

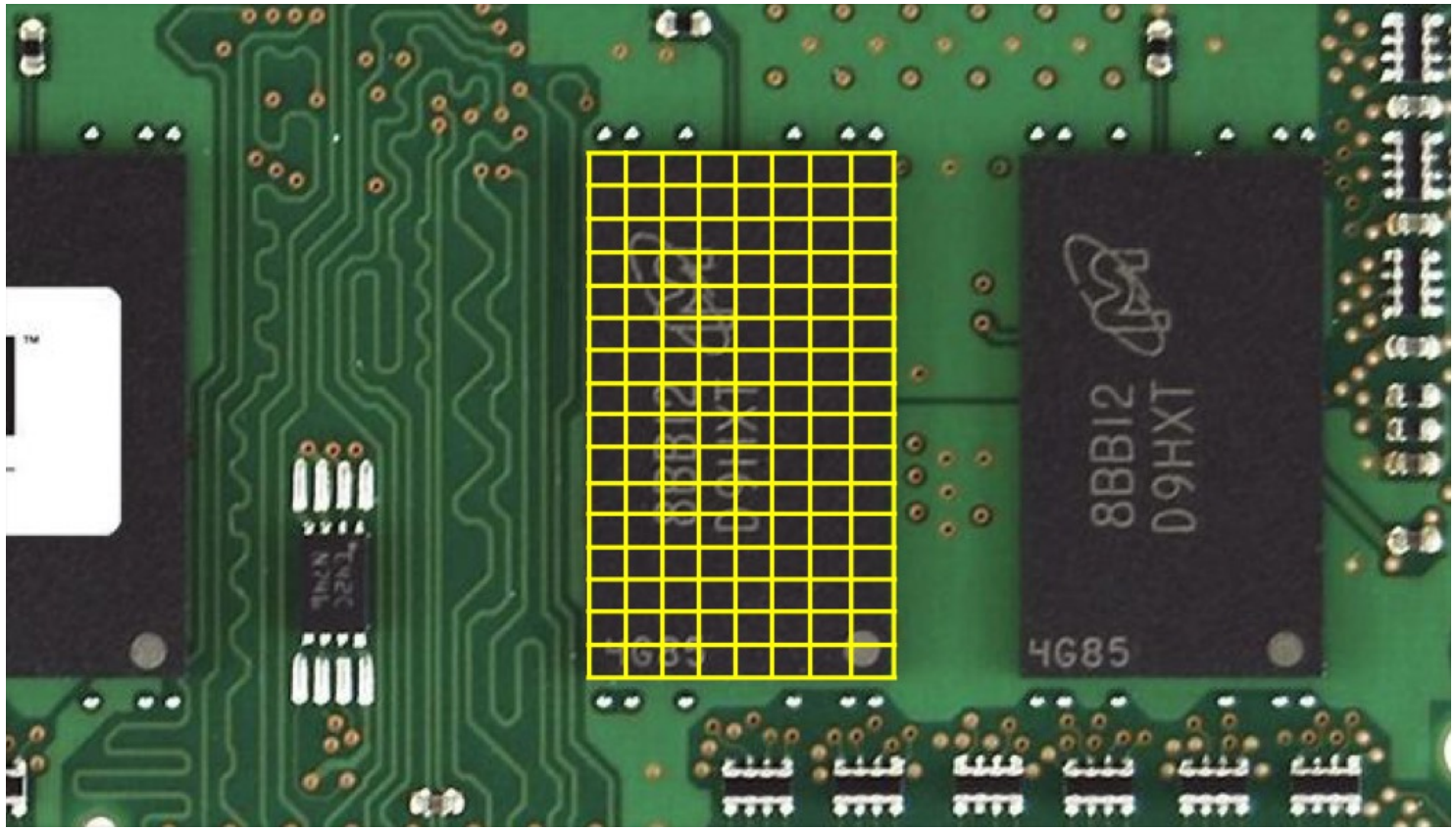
Memory


crucial
by Micron
47J6
4G85
8BB12
D9HXT
4G85
8BB12
D9HXT
4G85

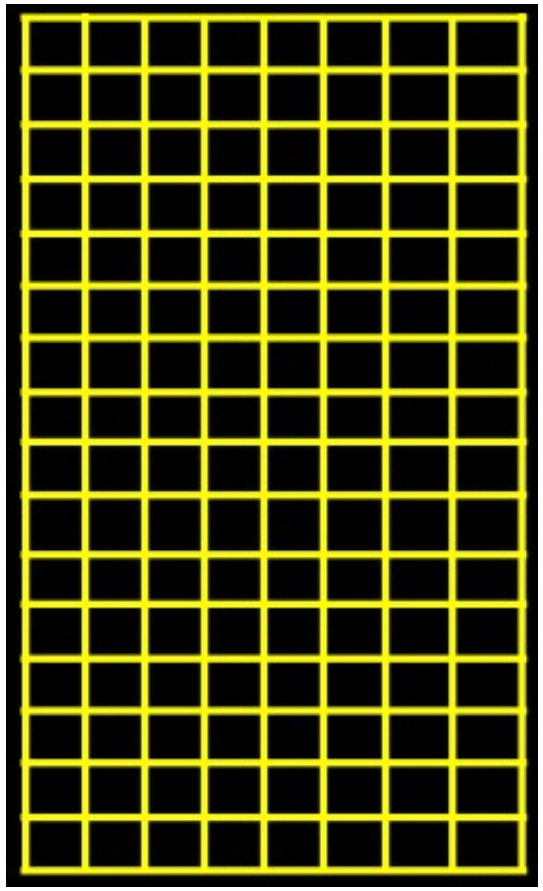
Memory


crucial
by Micron
8BB12
D9HXT
8BB12
D9HXT
47J6
4G85
4G85
4G85

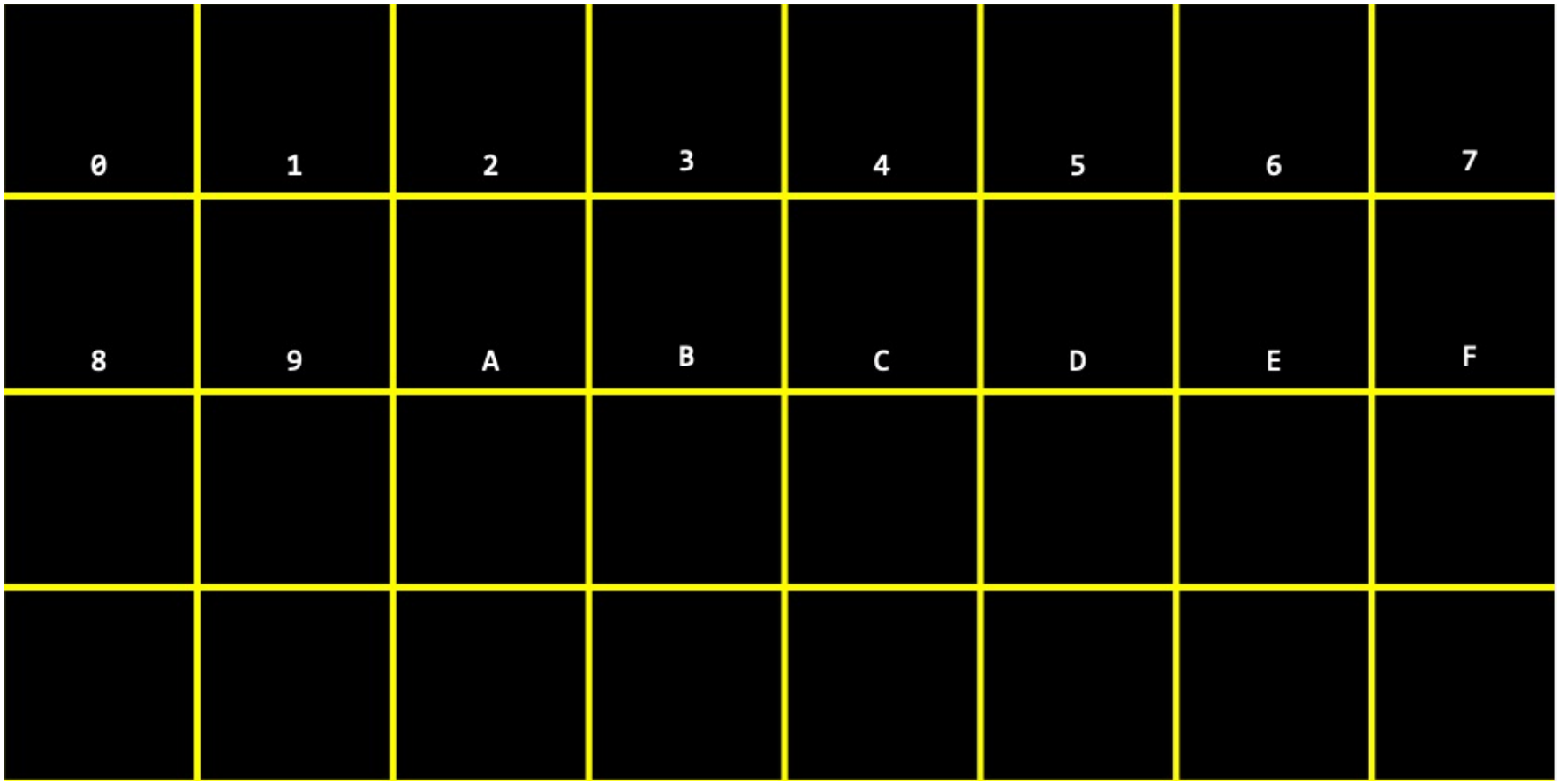
Memory



Memory



Memory



Pointers

&	Address Of operator
*	De-refence operator

Pointers

Tells us what address a variable is at?

& Address Of operator

* De-refence operator

Go to an address

Printing the value of an integer

```c
#include <stdio.h>

int main()
{
        // Declaration and initialization of
        // an int variable named n
        int n = 50;
        // Prints the variable's value
        printf("%i\n", n);

        return 0;
}
```

```
50
```

C Language

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

int main()
{
        // Declaration and initialization of
        // an int variable named n
        int n = 50;
        // Prints the variable's value
        cout << n << endl;

        return 0;
}
```

```
50
```

C++ Language

Printing the address of an integer

```
#include <stdio.h>

int main()
{
        // Declaration and initialization of
        // an int variable named n
        int n = 50;
        // Prints the variable's value
        printf("%p\n", &n);

        return 0;
}
```

```
0x7ffee7b6e798
```

C Language

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

int main()
{
        // Declaration and initialization of
        // an int variable named n
        int n = 50;
        // Prints the variable's value
        cout << &n << endl;

        return 0;
}
```

```
0x7ffee4a98798
```

C++ Language

Introducing Pointer Variable

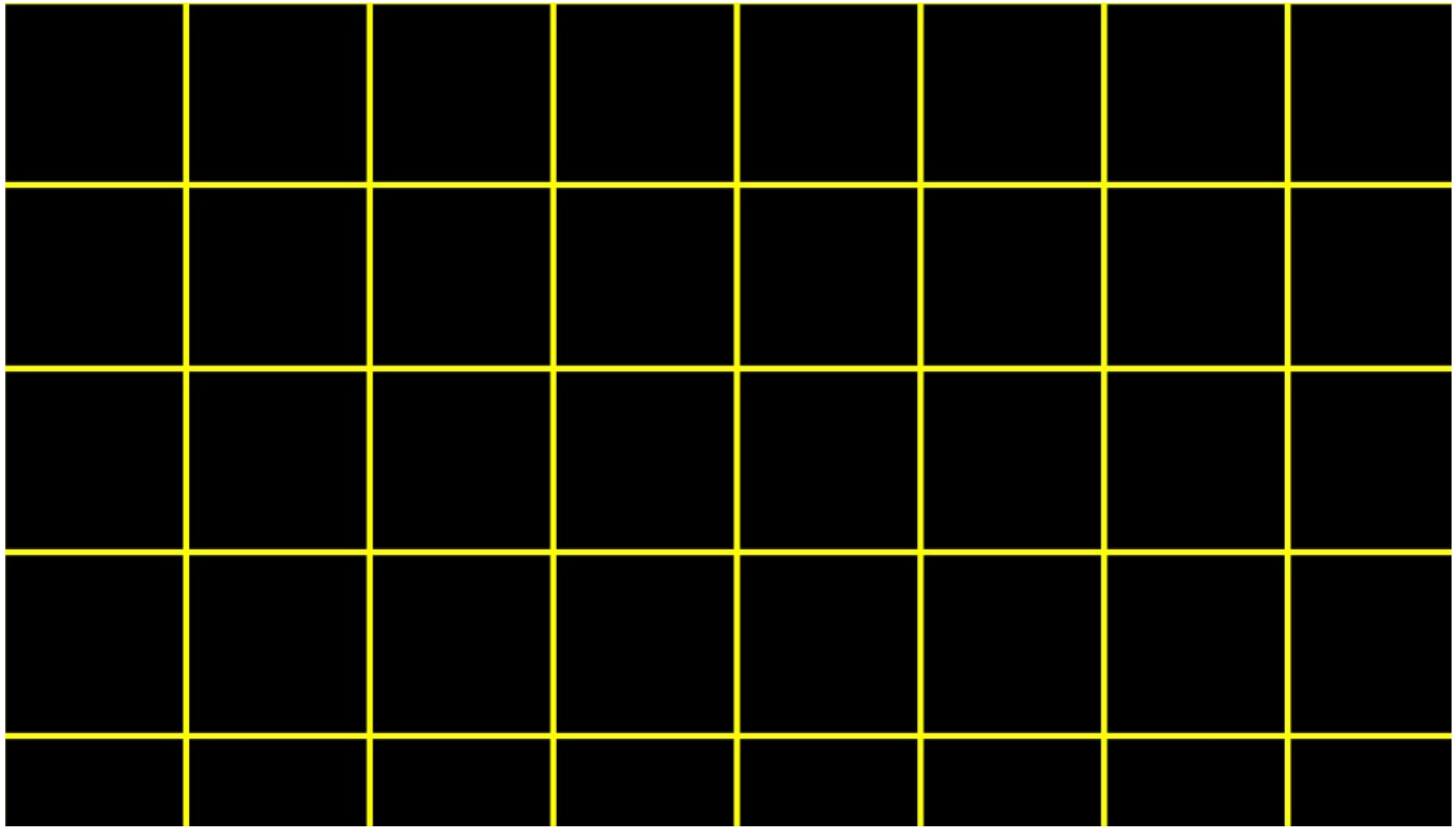
```c
#include <stdio.h>

int main()
{
        // Declaration and initialization of
        // an int variable named n
        int n = 50;
        // Pointer variable: 8-byte long
        // Points at the variable named n
        int *p = &n;

        return 0;
}
```

C Language

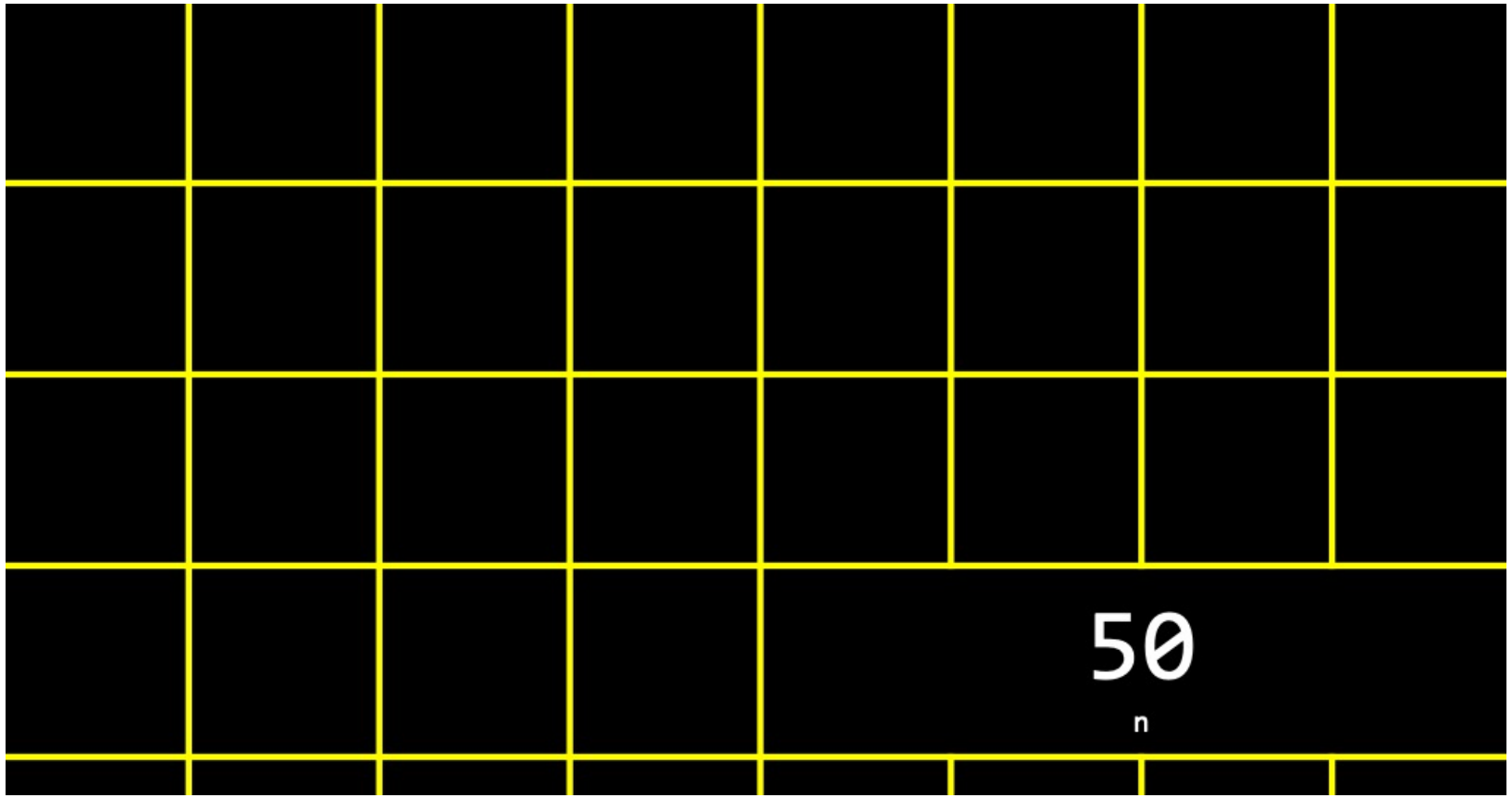
```cpp
#include <iostream>
using namespace std;

int main()
{
        // Declaration and initialization of
        // an int variable named n
        int n = 50;
        // Pointer variable: 8-byte long
        // Points at the variable named n
        int *p = &n;

        return 0;
}
```

C++ Language

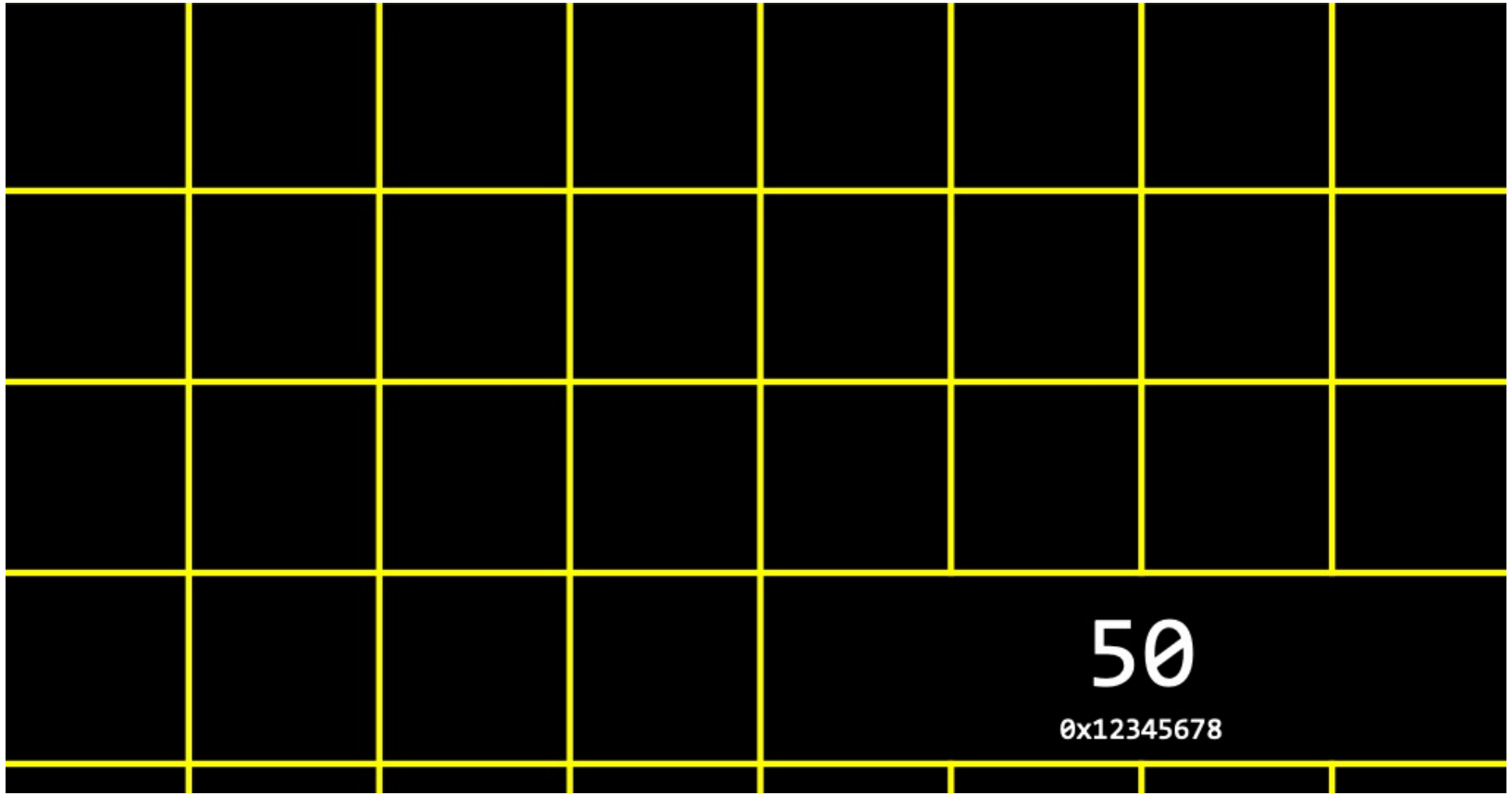
Introducing Pointer Variable



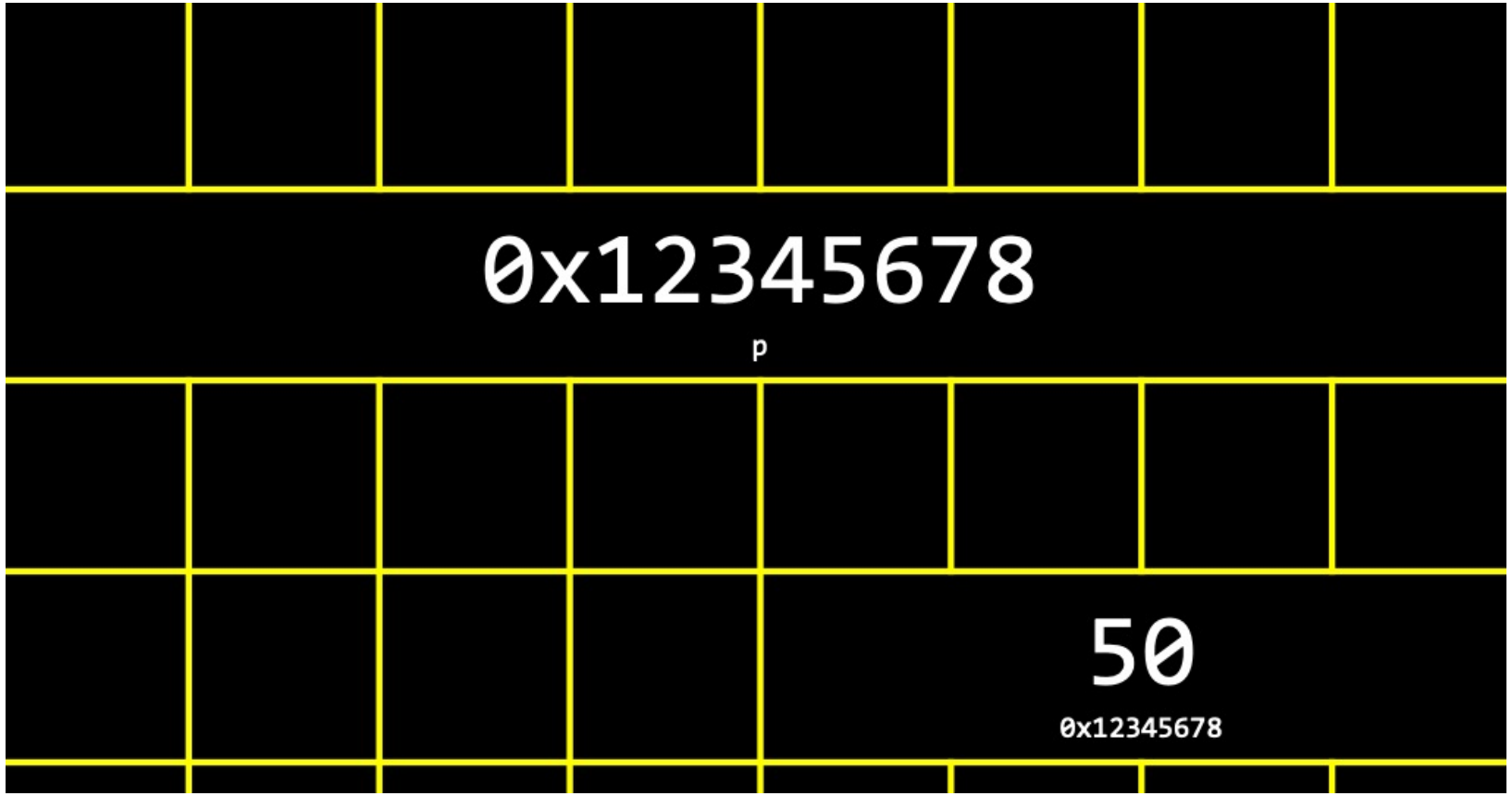
Introducing Pointer Variable



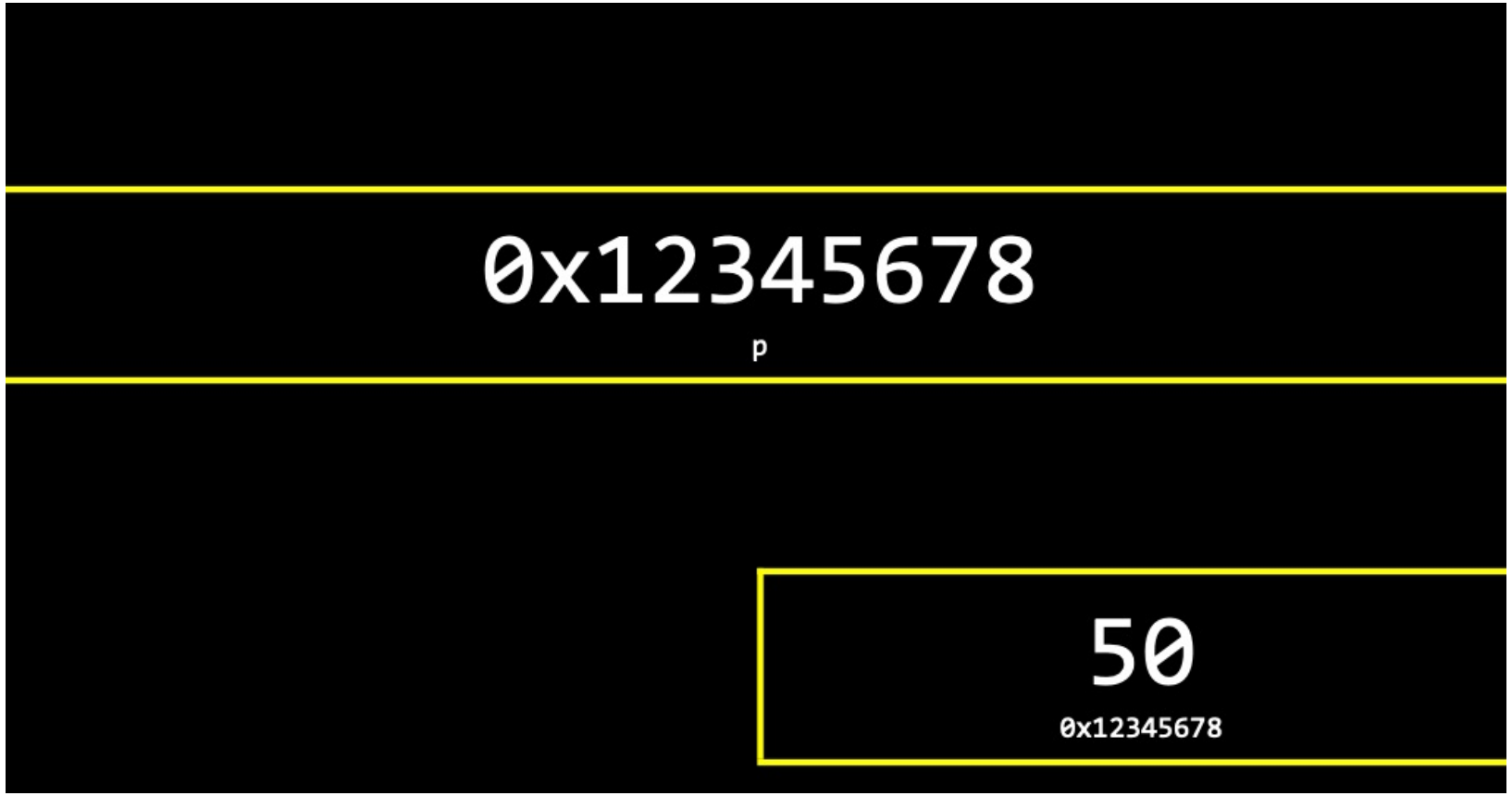
Introducing Pointer Variable



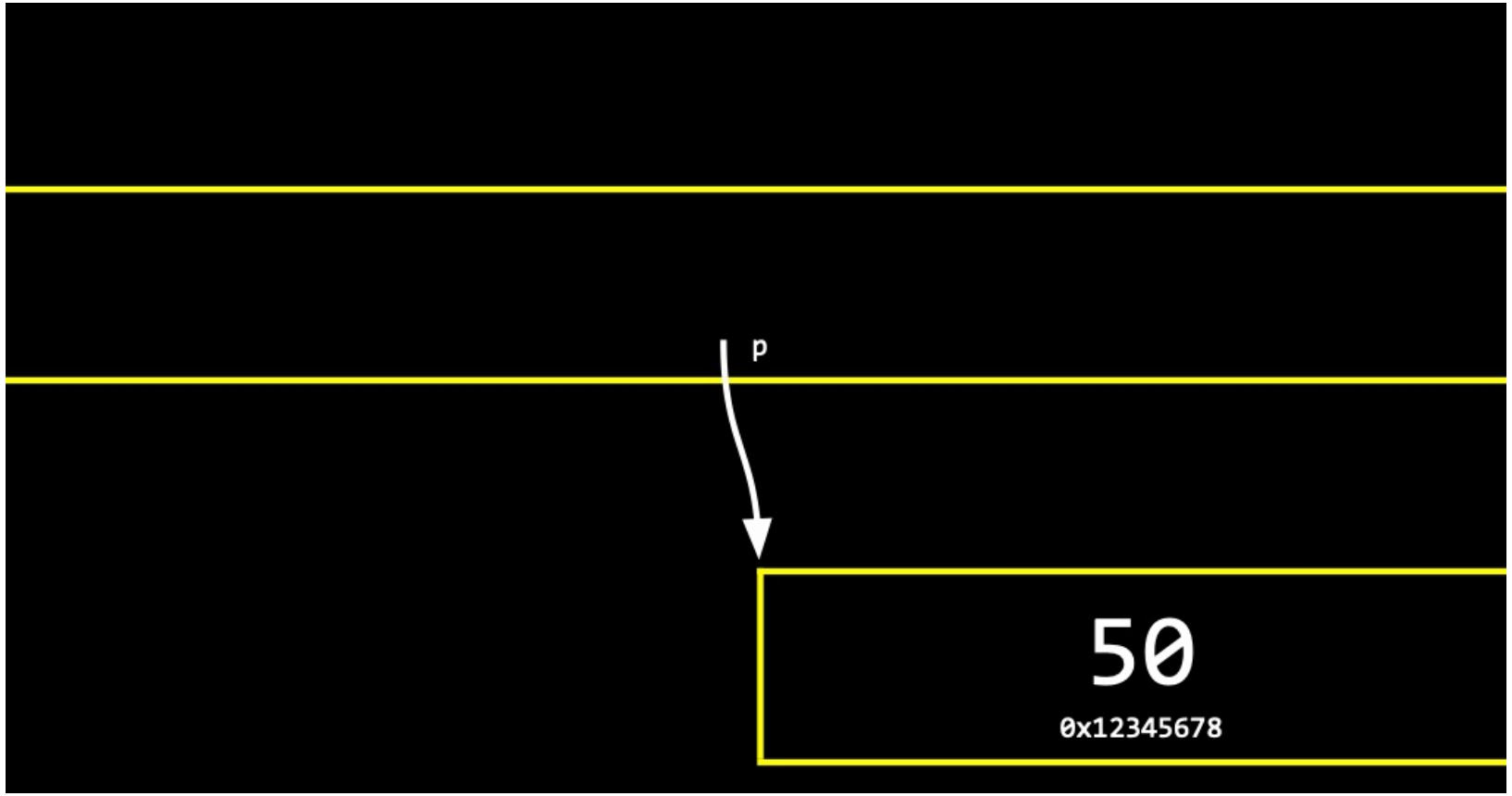
Introducing Pointer Variable



Introducing Pointer Variable



Introducing Pointer Variable



Printing the address of an integer using Pointer

```c
#include <stdio.h>

int main()
{
        // Declaration and initialization of
        // an int variable named n
        int n = 50;
        // Pointer variable: 8-byte long
        // Points at the variable named n
        int *p = &n;
        // Prints the address of the integer
        printf("%p\n", p);

        return 0;
}
```

```
0x7ffeeed17798
```

C Language

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

int main()
{
        // Declaration and initialization of
        // an int variable named n
        int n = 50;
        // Pointer variable: 8-byte long
        // Points at the variable named n
        int *p = &n;
        // Prints the address of the integer
        cout << p << endl;

        return 0;
}
```

```
0x7ffee1af8778
```

C++ Language

Printing the value of an integer using Pointer

```c
#include <stdio.h>

int main()
{
        // Declaration and initialization of
        // an int variable named n
        int n = 50;
        // Pointer variable: 8-byte long
        // Points at the variable named n
        int *p = &n;
        // Prints the value of the integer
        printf("%i\n", *p);

        return 0;
}
```

```
50
```

C Language

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

int main()
{
        // Declaration and initialization of
        // an int variable named n
        int n = 50;
        // Pointer variable: 8-byte long
        // Points at the variable named n
        int *p = &n;
        // Prints the value of the integer
        cout << *p << endl;

        return 0;
}
```

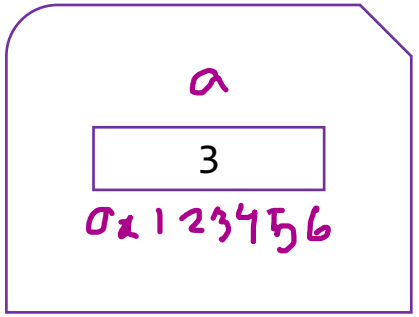
```
50
```

C++ Language

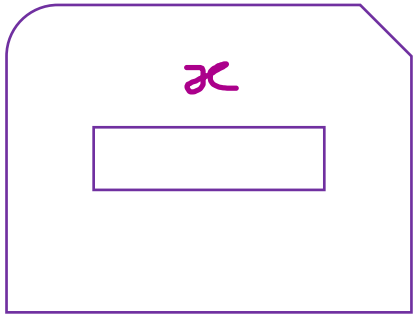
Pass by Reference

- To be used when we intend to modify the value of the original variable

Main function scope



Increment function scope



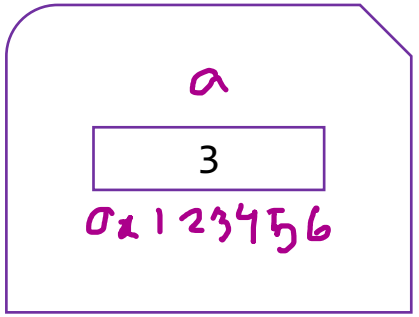

```
#include <iostream>
using namespace std;

void Increment(int *x)
{
        *x += 1;
        cout << “x in Increment: ” << *x << endl;
}

int main()
{
        int a = 3;
        Increment(&a);
        cout << “a in Main: ” << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 4
```

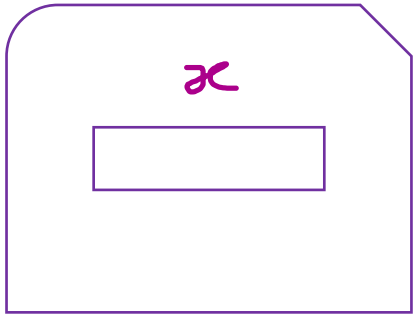
Pass by Reference

- To be used when we intend to modify the value of the original variable

Main function scope



Increment function scope

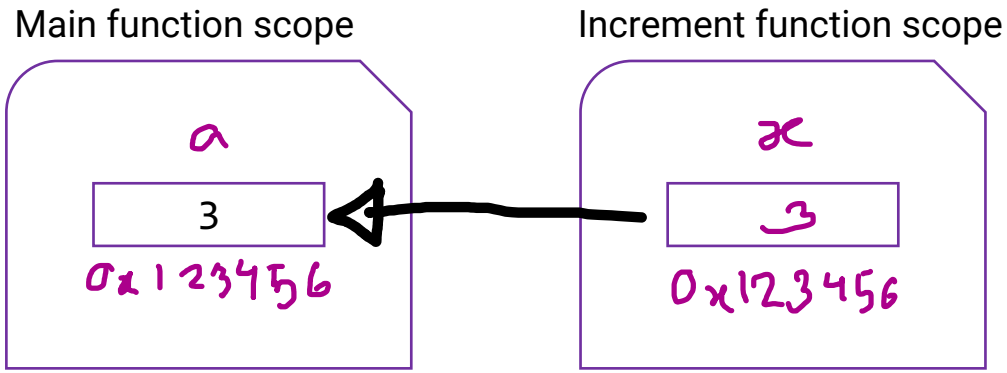



```
#include <iostream>
using namespace std;

void Increment(int *x)
{
        *x += 1;
        cout << “x in Increment: ” << *x << endl;
}

int main()
{
        int a = 3;
        Increment(&a);
        cout << “a in Main: ” << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 4
```

Pass by Reference

- To be used when we intend to modify the value of the original variable

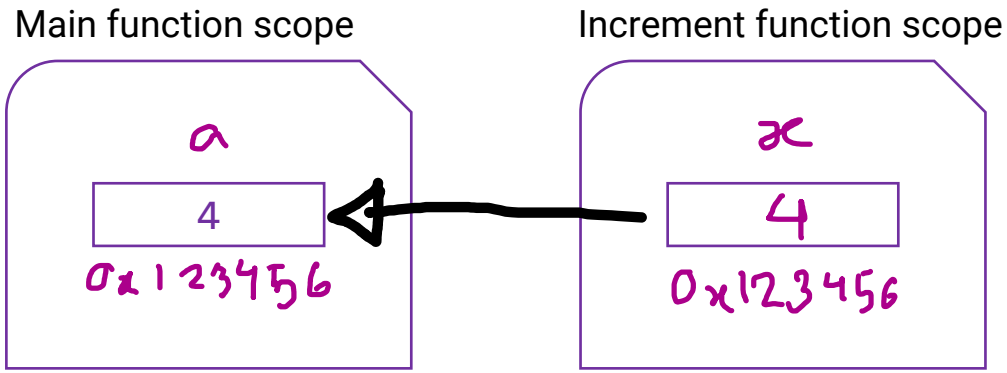



```
#include <iostream>
using namespace std;

void Increment(int *x)
{
        *x += 1;
        cout << “x in Increment: ” << *x << endl;
}

int main()
{
        int a = 3;
        Increment(&a);
        cout << “a in Main: ” << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 4
```

Pass by Reference

- To be used when we intend to modify the value of the original variable

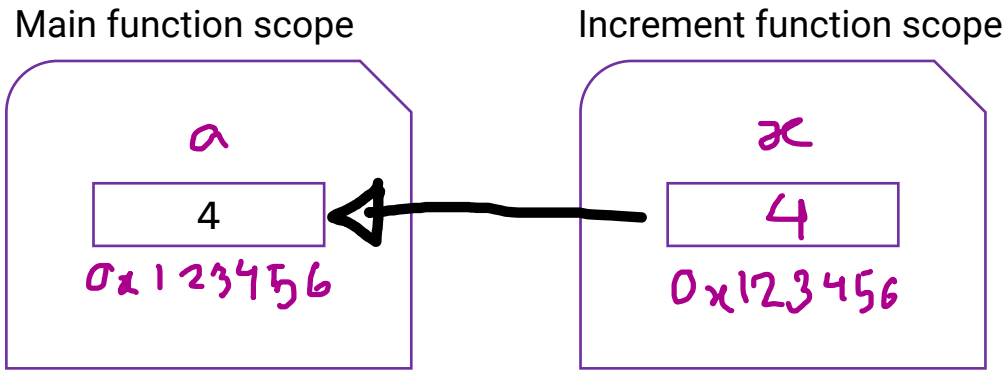



```
#include <iostream>
using namespace std;

void Increment(int *x)
{
        *x += 1;
        cout << “x in Increment: ” << *x << endl;
}

int main()
{
        int a = 3;
        Increment(&a);
        cout << “a in Main: ” << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 4
```

Pass by Reference

- To be used when we intend to modify the value of the original variable

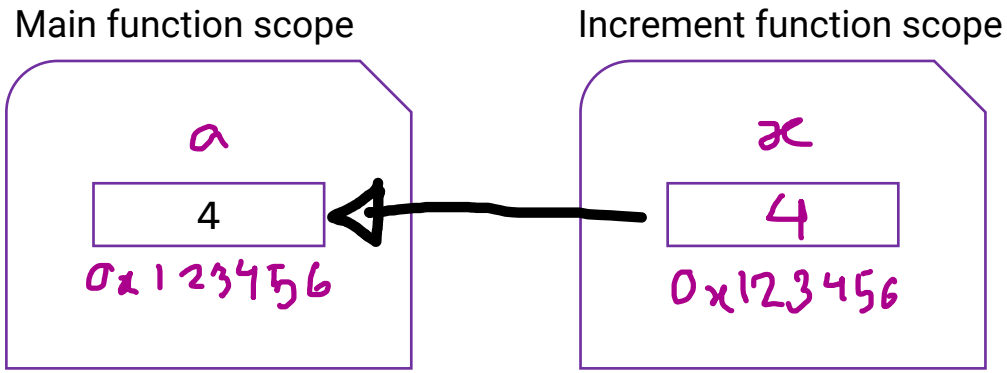



```
#include <iostream>
using namespace std;

void Increment(int *x)
{
        *x += 1;
        cout << “x in Increment: ” << *x << endl;
}

int main()
{
        int a = 3;
        Increment(&a);
        cout << “a in Main: ” << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 4
```

Pass by Reference

- To be used when we intend to modify the value of the original variable

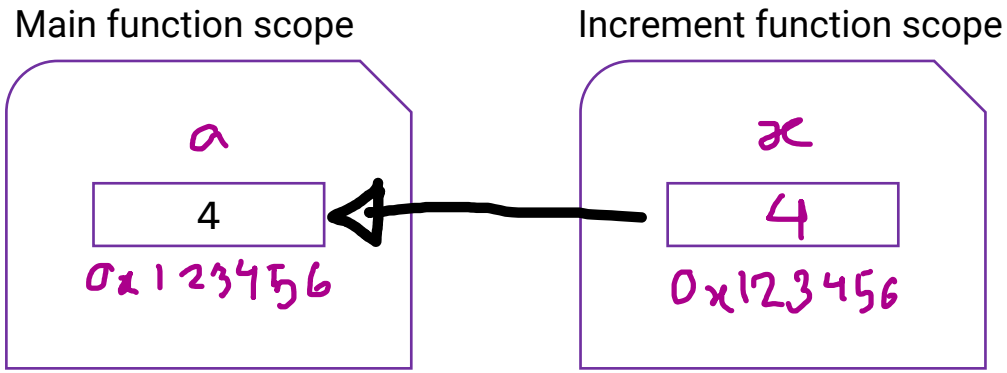



```
#include <iostream>
using namespace std;

void Increment(int *x)
{
        *x += 1;
        cout << “x in Increment: ” << *x << endl;
}

int main()
{
        int a = 3;
        Increment(&a);
        cout << “a in Main: ” << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 4
```

Pass by Reference

- To be used when we intend to modify the value of the original variable



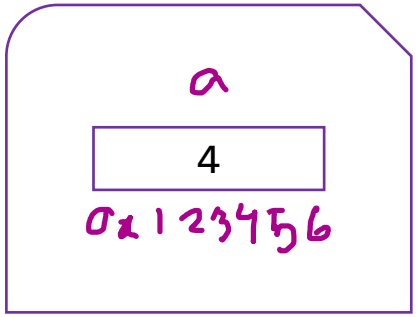

```
#include <iostream>
using namespace std;

void Increment(int *x)
{
        *x += 1;
        cout << “x in Increment: ” << *x << endl;
}

int main()
{
        int a = 3;
        Increment(&a);
        cout << “a in Main: ” << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 4
```

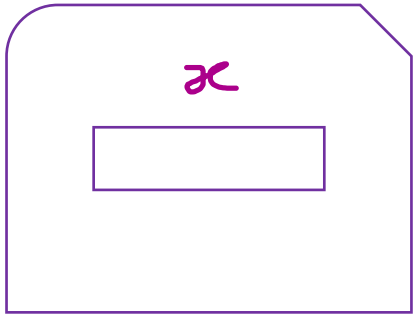
Pass by Reference

- To be used when we intend to modify the value of the original variable

Main function scope



Increment function scope



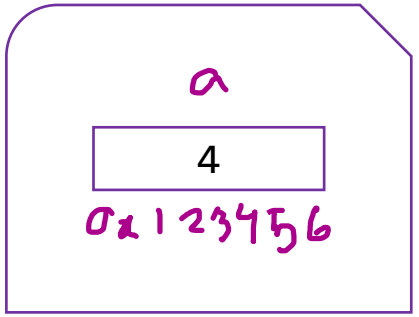

```
#include <iostream>
using namespace std;

void Increment(int *x)
{
        *x += 1;
        cout << “x in Increment: ” << *x << endl;
}

int main()
{
        int a = 3;
        Increment(&a);
        cout << “a in Main: ” << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 4
```

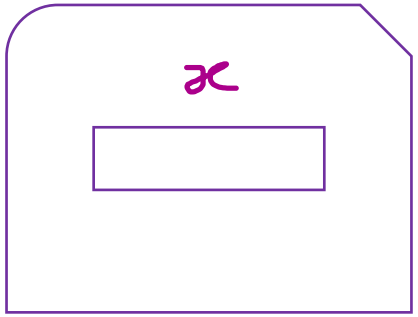
Pass by Reference

- To be used when we intend to modify the value of the original variable

Main function scope



Increment function scope




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

void Increment(int *x)
{
        *x += 1;
        cout << “x in Increment: ” << *x << endl;
}

int main()
{
        int a = 3;
        Increment(&a);
        cout << “a in Main: ” << a << endl;
        return 0;
}
```

```
x in Increment: 4
a in Main: 4
```

Output Tracing

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

Void Swap(int *x, int *y)
{
        int temp = *x;
        *x = *y;
        *y = temp;
}

int main()
{
        int a = 10, b = 20;
        cout << a << “ ” << b << endl;
        Swap(&a, &b);
        cout << a << “ ” << b << endl;
        return 0;
}
```

```
??
```

Output Tracing

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

Void Swap(int *x, int *y)
{
        int temp = *x;
        *x = *y;
        *y = temp;
}

int main()
{
        int a = 10, b = 20;
        cout << a << “ ” << b << endl;
        Swap(&a, &b);
        cout << a << “ ” << b << endl;
        return 0;
}
```

```
10 20
20 10
```

Programming Challenge

Write a C++ program that can compute the area of a circle (given the radius), the radius of a circle (given the area), the diameter of a circle (given the radius) using Functions.

Solution: Programming Challenge

Write a C++ program that can compute the area of a circle (given the radius), the radius of a circle (given the area), the diameter of a circle (given the radius) using Functions.

```
double Area (double radius)
{
        // Some code to return
        // the computed area
}
```

```
double Radius (double area)
{
        // Some code to return
        // the computed radius;
}
```

```
double Diameter (double radius)
{
        // Some code to return
        // the diameter
}
```

Solution: Programming Challenge

Write a C++ program that can compute the area of a circle (given the radius), the radius of a circle (given the area), the diameter of a circle (given the radius) using Functions.

```
double Area (double radius)
{
        // Some code to return
        // the computed area
}
```

```
double Area (double);
```

Prototype for the Area function

```
double Radius (double area)
{
        // Some code to return
        // the computed radius;
}
```

```
double Radius (double);
```

Prototype for the Radius function

```
double Diameter (double radius)
{
        // Some code to return
        // the diameter
}
```

```
double Diameter (double);
```

Prototype for the Diameter function

Solution: Programming Challenge

Write a C++ program that can compute the area of a circle (given the radius), the radius of a circle (given the area), the diameter of a circle (given the radius) using Functions.

Function prototypes are generally put into separate header files. It separates specification of the function from its implementation.

circle.h

```
double Area (double);
double Radius (double);
double Diameter (double);
```

Solution: Programming Challenge

Write a C++ program that can compute the area of a circle (given the radius), the radius of a circle (given the area), the diameter of a circle (given the radius) using Functions.

Function prototypes are generally put into separate header files. It separates specification of the function from its implementation.

circle.h

```
#ifndef CIRCLE_H
#define CIRCLE_H

// To compute the area of circle given the radius
double Area (double);
// To compute the radius of a circle given the area
double Radius (double);
// To compute the diameter of a circle given the radius
double Diameter (double);

#endif
```

Solution: Programming Challenge

Function prototypes are generally put into separate header files. It separates specification of the function from its implementation.

circle.h

```
#ifndef CIRCLE_H
#define CIRCLE_H

// To compute the area of circle given the radius
double Area (double);
// To compute the radius of a circle given the area
double Radius (double);
// To compute the diameter of a circle given the radius
double Diameter (double);

#endif
```

circle.cpp

```
#include <iostream>
#include <cmath>   // To perform the math operations
using namespace std;

double Area (double radius)
{
        return (3.1416 * radius * radius);
}

double Radius (double area)
{
        return sqrt(area / 3.1416);
}

double Diameter (double radius)
{
        return (2 * radius);
}
```

Solution: Programming Challenge

Function prototypes are generally put into separate header files. It separates specification of the function from its implementation.

circle.h

```
#ifndef CIRCLE_H
#define CIRCLE_H

// To compute the area of circle given the radius
double Area (double);
// To compute the radius of a circle given the area
double Radius (double);
// To compute the diameter of a circle given the radius
double Diameter (double);

#endif
```

circle.cpp

```
#include <iostream>
#include <cmath>   // To perform the math operations
using namespace std;

double Area (double radius)
{
        return (3.1416 * radius * radius);
}

double Radius (double area)
{
        return sqrt(area / 3.1416);
}

double Diameter (double radius)
{
        return (2 * radius);
}
```

Solution: Programming Challenge

main.cpp

```
#include <iostream>
#include “circle.h”   //Include the circle header file
using namespace std;

int main()
{
            double radius = 5.4;
            cout << “Area: ” << Area(radius) << endl;
            cout << “Dimeter: ” << Diameter(radius) << endl;

            double area = 35.26;
            cout << “Radius: ” << Radius(area) << endl;
            return 0;
}
```

circle.h

```
#ifndef CIRCLE_H
#define CIRCLE_H

// To compute the area of circle given the radius
double Area (double);
// To compute the radius of a circle given the area
double Radius (double);
// To compute the diameter of a circle given the radius
double Diameter (double);

#endif
```

circle.cpp

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

double Area (double radius)
{
            return (3.1416 * radius * radius);
}
double Radius (double area)
{
            return sqrt(area / 3.1416);
}
double Diameter (double radius)
{
            return (2 * radius);
}
```

Solution: Programming Challenge

main.cpp

```
#include <iostream>
#include “circle.h”   //Include the circle header file
using namespace std;

int main()
{
            double radius = 5.4;
            cout << “Area: ” << Area(radius) << endl;
            cout << “Dimeter: ” << Diameter(radius) << endl;

            double area = 35.26;
            cout << “Radius: ” << Radius(area) << endl;
            return 0;
}
```

Terminal / Command Prompt

```
$ g++ -o main main.cpp circle.cpp
$./main
Area: 91.6091
Diameter: 10.8
Radius: 3.35016
```

circle.h

```
#ifndef CIRCLE_H
#define CIRCLE_H

// To compute the area of circle given the radius
double Area (double);
// To compute the radius of a circle given the area
double Radius (double);
// To compute the diameter of a circle given the radius
double Diameter (double);

#endif
```

circle.cpp

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

double Area (double radius)
{
            return (3.1416 * radius * radius);
}
double Radius (double area)
{
            return sqrt(area / 3.1416);
}
double Diameter (double radius)
{
            return (2 * radius);
}
```

Remarks

- Reference
  - Functions, Programiz. https://www.programiz.com/cpp-programming/function
  - MIT 6.096 Introduction to C++
  - This is CS50x, Dr. David J. Malan. Week 4: Memory. https://cs50.harvard.edu/x/2020/
  - Kanetkar, Yashavant P. "Let Us C."