

6.2 Return

Returning a value from a function

A function may return one value using a **return statement**. Below, the ComputeSquare() function is defined to have a return type of int; thus, the function's return statement must have an expression that evaluates to an int.

PARTICIPATION ACTIVITY

6.2.1: Function returns computed square.

[Start](#)☐ 2x speed

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

int ComputeSquare(int numToSquare) {
    return numToSquare * numToSquare;
}

int main() {
    int numSquared;

    numSquared = ComputeSquare(7);
    cout << "7 squared is " << numSquared << endl;

    return 0;
}
```

7 squared is 49

[Feedback?](#)

Other return types are allowed, such as char, double, etc. A function can only return one item, not two or more. A return type of **void** indicates that a function does not return any value.

PARTICIPATION ACTIVITY

6.2.2: Return.

Given the definition below, indicate which are valid return statements:

```
int CalculateSomeValue(int num1, int num2) { ... }
```

1) return 9;

☐ Yes

☐

No

2) `return num1;`

☐ Yes

☐ No

3) `return (num1 + num2) + 1 ;`

☐ Yes

☐ No

4) `return;`

☐ Yes

☐ No

5) `return num1 num2;`

☐ Yes

☐ No

6) `return (0);`

☐ Yes

☐ No

[Feedback?](#)

**PARTICIPATION
ACTIVITY**

6.2.3: More on return.

1) The following is a valid function definition:

```
char GetItem() { ... }
```

☐ True

☐ False

2) The following is a valid function definition for a function that returns two items:

```
int, int GetItems() { ... }
```

☐ True

☐ False

- 3) The following is a valid function definition:

```
void PrintItem() { ... }
```

- ☐ True
☐ False

[Feedback?](#)

Calling functions in expressions

A function call evaluates to the returned value. Thus, a function call often appears within an expression. Ex: `5 + ComputeSquare(4)` evaluates to `5 + 16`, or `21`.

A function with a void return type cannot be used within an expression, instead being used in a statement like: `OutputData(x, y);`

PARTICIPATION ACTIVITY

6.2.4: Calls in an expression.

Given the definitions below, which are valid statements?

```
double SquareRoot(double x) { ... }  
void PrintVal(double x) { ... }  
double y;
```

- 1) `y = SquareRoot(49.0);`
☐ True
☐ False
- 2) `SquareRoot(49.0) = z;`
☐ True
☐ False
- 3) `y = 1.0 + SquareRoot(144.0);`
☐ True
☐ False
- 4) `y = SquareRoot(SquareRoot(16.0));`
☐ True
☐ False
- 5) `y = SquareRoot();`

☐ True

☐ False

6) `SquareRoot(9.0);`



☐ True

☐ False

7) `y = PrintVal(9.0);`



☐ True

☐ False

8) `PrintVal(9.0);`



☐ True

☐ False

[Feedback?](#)

Mathematical functions

A function is commonly defined to compute a mathematical function involving several numerical parameters and returning a numerical result. The program below uses a function to convert a person's height in U.S. units (feet and inches) into total centimeters.

Figure 6.2.1: Program with a function to convert height in feet/inches to centimeters.

```
Enter feet: 5
Enter inches: 8
Centimeters: 172.72
```

```

#include <iostream>
using namespace std;

/* Converts a height in feet/inches to centimeters */
double HeightFtInToCm(int heightFt, int heightIn) {
    const double CM_PER_IN = 2.54;
    const int    IN_PER_FT = 12;
    int totIn;
    double cmVal;

    totIn = (heightFt * IN_PER_FT) + heightIn; // Total inches
    cmVal = totIn * CM_PER_IN;                // Conv inch to cm
    return cmVal;
}

int main() {
    int userFt; // User defined feet
    int userIn; // User defined inches

    // Prompt user for feet/inches
    cout << "Enter feet: ";
    cin >> userFt;

    cout << "Enter inches: ";
    cin >> userIn;

    // Output the conversion result
    cout << "Centimeters: ";
    cout << HeightFtInToCm(userFt, userIn) << endl;

    return 0;
}

```

[Feedback?](#)

Human average height is increasing, attributed to better nutrition. Source: *Our World in Data: Human height*.

PARTICIPATION ACTIVITY

6.2.5: Mathematical functions.

Indicate which is a valid use of the HeightFtInToCm() function above. x is type double.

1) $x = \text{HeightFtInToCm}(5, 0);$

- ☐ Valid
☐ Not valid

2) $x = 2 * (\text{HeightFtInToCm}(5, 0) + 1.0);$

☐

Valid

☐ Not valid

3) $x = (\text{HeightFtInToCm}(5, 0) + \text{HeightFtInToCm}(6, 1)) / 2.0;$

☐ Valid

☐ Not valid

4) Suppose `int pow(int y, int z)` returns `y` to the power of `z`. Is the following valid?

`x = pow(2, pow(3, 2));`

☐ Valid

☐ Not valid

[Feedback?](#)

zyDE 6.2.1: Temperature conversion.

Complete the program by writing and calling a function that converts a temperature in Celsius into Fahrenheit.

[Load default template...](#)

```
1 #include <iostream>
2 using namespace std;
3
4
5 // FINISH: Define CelsiusToFahrenheit function
6
7
8 int main() {
9     double tempF;
10    double tempC;
11
12    cout << "Enter temperature in Celsius: " << endl;
13    cin >> tempC;
14
15    tempF = 0.0; // FIXME
16
17    cout << "Fahrenheit: " << tempF;
18
19    return 0;
20 }
21
```

100

Run

[Feedback?](#)

Calling functions from functions

A function's statements may call other functions. In the example below, the `PizzaCalories()` function calls the `CalcCircleArea()` function. (Note that `main()` itself is the first function called when a program executes, and calls other functions.)

Figure 6.2.2: Functions calling functions.

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

double CalcCircleArea(double circleDiameter) {
    double circleRadius;
    double circleArea;
    double piVal = 3.14159265;

    circleRadius = circleDiameter / 2.0;
    circleArea = piVal * circleRadius * circleRadius;

    return circleArea;
}

double PizzaCalories(double pizzaDiameter) {
    double totalCalories;
    double caloriesPerSquareInch = 16.7;    // Regular crust pepperoni pizza

    totalCalories = CalcCircleArea(pizzaDiameter) * caloriesPerSquareInch;

    return totalCalories;
}

int main() {
    cout << "12 inch pizza has " << PizzaCalories(12.0) << " calories." << endl;
    cout << "14 inch pizza has " << PizzaCalories(14.0) << " calories." << endl;

    return 0;
}
```

```
12 inch pizza has 1888.73 calories.
14 inch pizza has 2570.77 calories.
```

[Feedback?](#)

PARTICIPATION ACTIVITY

6.2.6: Functions calling functions.



Complete the `PizzaCaloriesPerSlice()` function to compute the calories for a single slice of pizza. A `PizzaCalories()` function returns a pizza's total calories given the pizza diameter passed as an argument. A `PizzaSlices()` function returns the number of slices in a pizza given the pizza diameter passed as an argument.

```
double PizzaCaloriesPerSlice(double pizzaDiameter) {  
    double totalCalories;  
    double caloriesPerSlice;  
  
    totalCalories = Placeholder_A;  
    caloriesPerSlice = Placeholder_B;  
  
    return caloriesPerSlice;  
}
```

- 1) Type the expression for Placeholder_A to compute the total calories for a pizza with diameter pizzaDiameter.

totalCalories =
 ;

Check

[Show answer](#)

- 2) Type the expression for Placeholder_B to compute the calories per slice.

caloriesPerSlice =

;

Check

[Show answer](#)

[Feedback?](#)

Exploring further:

- [Function definition](#) from msdn.microsoft.com
- [Function call](#) from msdn.microsoft.com

**CHALLENGE
ACTIVITY**

6.2.1: Enter the output of the returned value.



[Jump to level 1](#)

Type the program's output.


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

int ChangeValues(int x, int y) {
    int newValue;

    newValue = x + y;

    return newValue;
}

int main() {
    cout << ChangeValues(4, 3);

    return 0;
}
```

7

1

2

Check

Next

Done. Click any level to practice more. Completion is preserv

Yours

7

Expected

7

[Feedback?](#)**CHALLENGE
ACTIVITY**

6.2.2: Function call in expression.



Assign to maxSum the max of (numA, numB) PLUS the max of (numY, numZ). Use just one statement. Hint: Call FindMax() twice in an expression.

```
16 }
17
18 int main() {
19     double numA;
20     double numB;
21     double numY;
22     double numZ;
23     double maxSum;
24
25     cin >> numA;
26     cin >> numB;
27     cin >> numY;
28     cin >> numZ;
29
30     /* Your solution goes here */
31     maxSum = FindMax(numA, numB)+ FindMax(numY, numZ);
32
33     cout << "maxSum is: " << maxSum << endl;
34 }
```

```

35     return 0;
36 }

```

Run

✓ All tests passed

✓ Testing with inputs: 5.0 10.0 3.0 7.0

Your output

maxSum is: 17

✓ Testing with inputs: 1.0 2.0 25.3 24.6

Your output

maxSum is: 27.3

✓ Testing with inputs: -5.5 -99.32 0.0 0.5

Your output

maxSum is: -5

✓ Testing with inputs: 199.6 212.0 -2.0 -3.0

Your output

maxSum is: 210

✓ Testing with inputs: 0 0 1 1

Your output

maxSum is: 1

[Feedback?](#)**CHALLENGE
ACTIVITY**

6.2.3: Function definition: Volume of a pyramid.



Define a function `PyramidVolume` with double parameters `baseLength`, `baseWidth`, and `pyramidHeight`, that returns as a double the volume of a pyramid with a rectangular base. Relevant geometry equations:

Volume = base area x height x 1/3

Base area = base length x base width.

(Watch out for integer division).

```

1 #include <iostream>
2 using namespace std;

```

```
3  /* Your solution goes here */
4  double PyramidVolume(double baseLength, double baseWidth, double pyramidHeight){
5      return baseLength*baseWidth*pyramidHeight/3;
6  }
7
8  int main() {
9      double userLength;
10     double userWidth;
11     double userHeight;
12
13     cin >> userLength;
14     cin >> userWidth;
15     cin >> userHeight;
16
17     cout << "Volume: " << PyramidVolume(userLength, userWidth, userHeight) << endl;
18
19     return 0;
20 }
21
```

Run

✓ All tests passed

✓ Testing with inputs: 1.00 1.00 1.00

Your value

0.3333333333333333

✓ Testing with inputs: 5.80 4.00 6.00

Value differs. See highlights below.

Your value

46.4

[Feedback?](#)