6.1 User-defined function basics

Basics of functions

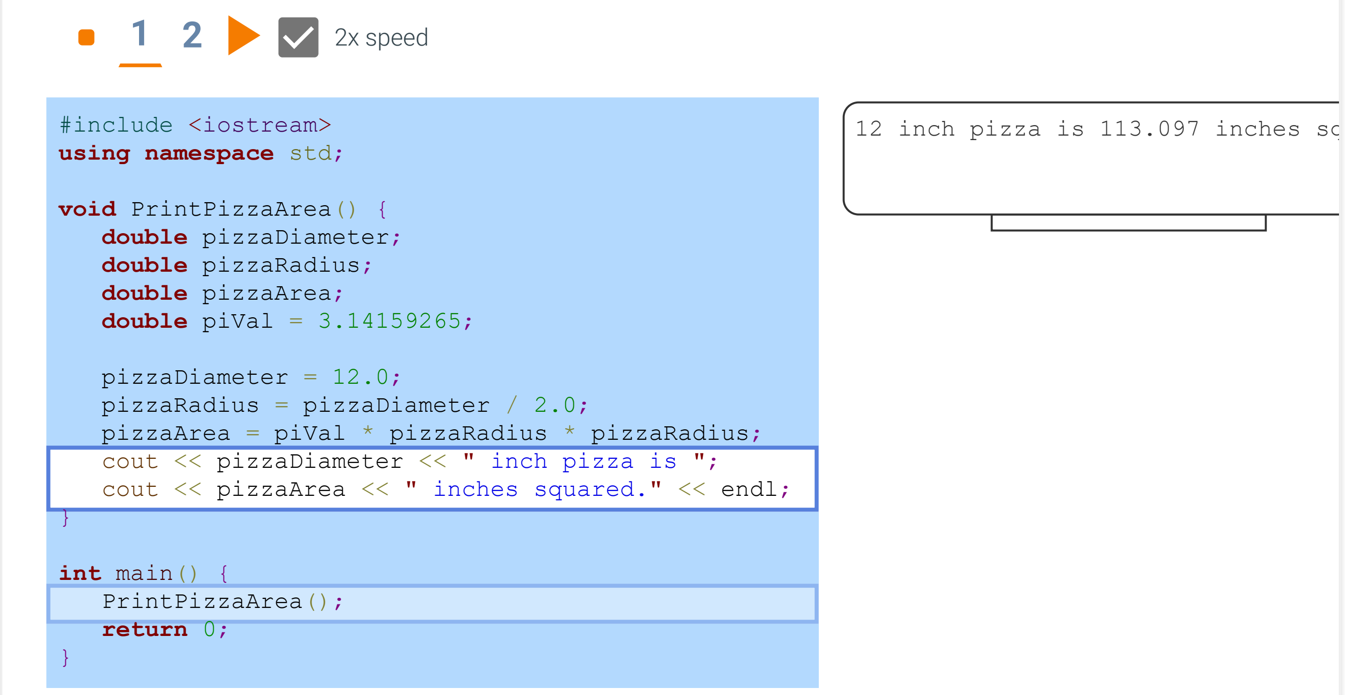
A ***function*** is a named list of statements.

- A ***function definition*** consists of the new function's name and a block of statements. Ex: `void PrintPizzaArea() { /* block of statements */ }`
- A ***function call*** is an invocation of a function's name, causing the function's statements to execute.

The function's name can be any valid identifier. A ***block*** is a list of statements surrounded by braces.

Below, the function call PrintPizzaArea() causes execution to jump to the function's statements. Execution returns to the original location after executing the function's last statement.

PARTICIPATION ACTIVITY 6.1.1: Function example: Printing a pizza area.



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

void PrintPizzaArea() {
   double pizzaDiameter;
   double pizzaRadius;
   double pizzaArea;
   double piVal = 3.14159265;

   pizzaDiameter = 12.0;
   pizzaRadius = pizzaDiameter / 2.0;
   pizzaArea = piVal * pizzaRadius * pizzaRadius;
   cout << pizzaDiameter << " inch pizza is ";
   cout << pizzaArea << " inches squared." << endl;
}

int main() {
   PrintPizzaArea();
   return 0;
}
```

```
12 inch pizza is 113.097 inches sq
```

The function call jumps execution to the function's statements.

Feedback?

PARTICIPATION ACTIVITY

6.1.2: Function basics.

Given the PrintPizzaArea() function defined above and the following main() function:

```
int main() {
   PrintPizzaArea();
   PrintPizzaArea();
   return 0;
}
```

1) How many function calls to PrintPizzaArea() exist in main()?

Check Show answer

2) How many function definitions of PrintPizzaArea() exist *within* main()?

Check Show answer

3) How many output statements would execute in total?

Check Show answer

4) How many output statements exist in PrintPizzaArea()?

Check Show answer

5) Is main() itself a function definition? Answer yes or no.

Check Show answer

Feedback?

Parameters

A programmer can influence a function's behavior via an input.

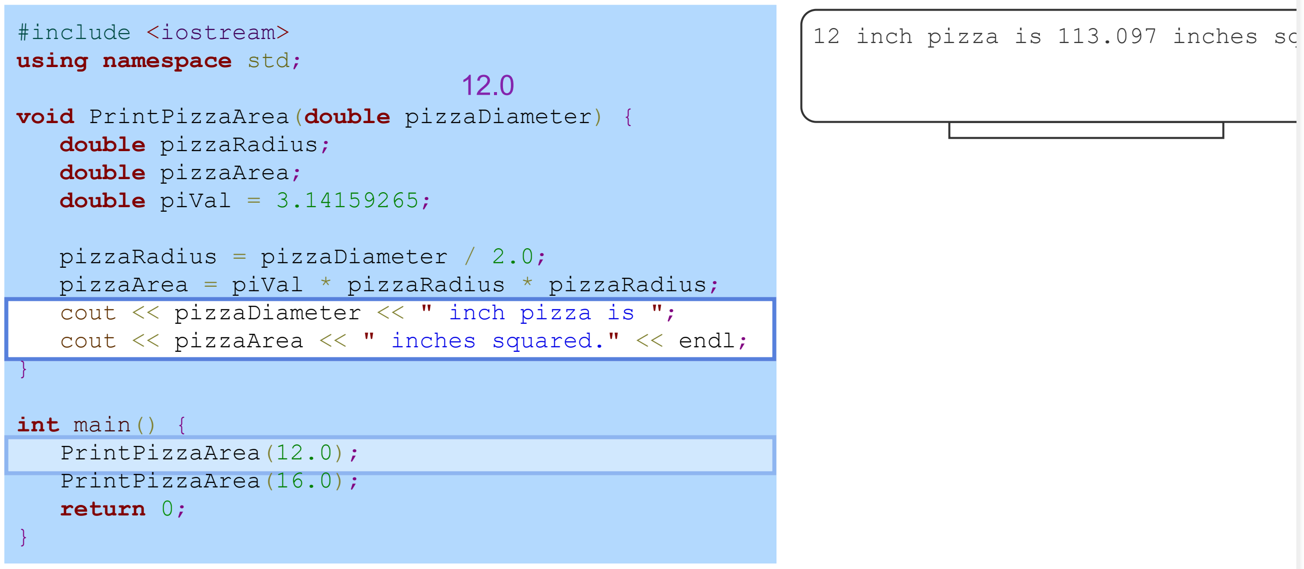
- A ***parameter*** is a function input specified in a function definition. Ex: A pizza area function might have diameter as an input.
- An ***argument*** is a value provided to a function's parameter during a function call. Ex: A pizza area function might be called as PrintPizzaArea(12.0) or as PrintPizzaArea(16.0).

A parameter is like a variable declaration. Upon a call, the parameter's memory location is allocated, and the parameter is assigned with the argument's value. Upon returning to the original call location, the parameter is deleted from memory.

An argument may be an expression, like 12.0, x, or x * 1.5.

PARTICIPATION ACTIVITY | 6.1.3: Function with parameters example: Printing a pizza area for different diameters.

1 2 2x speed



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

void PrintPizzaArea(double pizzaDiameter) {
   double pizzaRadius;
   double pizzaArea;
   double piVal = 3.14159265;

   pizzaRadius = pizzaDiameter / 2.0;
   pizzaArea = piVal * pizzaRadius * pizzaRadius;
   cout << pizzaDiameter << " inch pizza is ";
   cout << pizzaArea << " inches squared." << endl;
}

int main() {
   PrintPizzaArea(12.0);
   PrintPizzaArea(16.0);
   return 0;
}
```

```
12 inch pizza is 113.097 inches sq
```

The function call jumps execution to the function's statements, passing 12.0 to the function's pizzaDiameter parameter.

Feedback?

PARTICIPATION ACTIVITY | 6.1.4: Parameters.

1) Complete the function beginning to

have a parameter named userAge of type int.

```
void PrintAge(
) {
```

Check Show answer

2) Write a statement that calls a function named PrintAge, passing the value 21 as an argument.

Check Show answer

3) Is the following a valid function definition beginning? Type yes or no.

```
void MyFct(int userNum + 5)
{ ... }
```

Check Show answer

4) Assume a function `void PrintNum(int userNum)` simply prints the value of userNum without any space or new line. What will the following output?

```
PrintNum(43);
PrintNum(21);
```

Check Show answer

Feedback?

Multiple or no parameters

A function definition may have multiple parameters, separated by commas. Parameters are assigned with argument values by position: First parameter with first argument, second with second, etc.

A function definition with no parameters must still have the parentheses, as in: `void PrintSomething() { ... }`. The call must include parentheses, with no argument, as in: `PrintSomething()`.

Figure 6.1.1: Function with multiple parameters.

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

void PrintPizzaVolume(double pizzaDiameter, double pizzaHeight) {
   double pizzaRadius;
   double pizzaArea;
   double pizzaVolume;
   double piVal = 3.14159265;

   pizzaRadius = pizzaDiameter / 2.0;
   pizzaArea = piVal * pizzaRadius * pizzaRadius;
   pizzaVolume = pizzaArea * pizzaHeight;
   cout << pizzaDiameter << " x " << pizzaHeight << " inch pizza is ";
   cout << pizzaVolume << " inches cubed." << endl;
}

int main() {
   PrintPizzaVolume(12.0, 0.3);
   PrintPizzaVolume(12.0, 0.8);
   PrintPizzaVolume(16.0, 0.8);
   return 0;
}
```

```
12 x 0.3 inch pizza is 33.9292 inches cubed.
12 x 0.8 inch pizza is 90.4779 inches cubed.
16 x 0.8 inch pizza is 160.85 inches cubed.
```

Feedback?

PARTICIPATION ACTIVITY 6.1.5: Multiple parameters.

1) Which correctly defines two integer parameters x and y for a function definition:
`void CalcVal(...)`?

- ○ (int x; int y)
- ○ (int x, y)
- ○ (int x, int y)

2) Which correctly passes two integer arguments for the function call:
`CalcVal(...)`?

- ○ (99, 44 + 5)
- ○ (int 99, 44)
- ○ (int 99, int 44)

3) Given a function definition:
`void CalcVal(int a, int b, int c)`
b is assigned with what value during this function call:
`CalcVal(42, 55, 77);`

- ○ Unknown
- ○ 42
- ○ 55

4) Given a function definition:
`void CalcVal(int a, int b, int c)`
and given int variables i, j, and k, which are valid arguments in the call `CalcVal(...)`?

- ○ (i, j)
- ○ (k, i + j, 99)
- ○ (i + j + k)

Feedback?

PARTICIPATION ACTIVITY 6.1.6: Calls with multiple parameters.

Given:

```
void PrintSum(int num1, int num2) {
   cout << num1 << " + " << num2 << " is " << (num1 + num2);
}
```

1) What will be printed for the following function call?
`PrintSum(1, 2);`

Check Show answer

2) Write a statement that calls PrintSum() to print the sum of x and 400 (providing the arguments in that order). End with ;

Check Show answer

Feedback?

Exploring further:

- Functions tutorial from cplusplus.com

CHALLENGE ACTIVITY 6.1.1: Function parameters.

Jump to level 1

Type the program's output.

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

void printPoints(string name, int age, int totalPoints) {
   cout << name << " is " << age << endl;
   cout << name << " made " << totalPoints << " points" << endl;
}

int main() {
   string userName = "Bob";
   int userAge = 19;
   int regularTimePoints = 25;
   int overtimePoints = 3;

   printPoints(userName, userAge, regularTimePoints + overtimePoints);

   return 0;
}
```

```
Bob i
Bob m
```

1 2 3 4

Check Next **Done**. Click any level to practice more. Completion is preserv

✓ printPoints is called, and userName, userAge, and the evaluated result of regularTimePoint

Yours

```
Bob is 19
Bob made 28 points
```

Expected

```
Bob is 19
Bob made 28 points
```

Feedback?

CHALLENGE ACTIVITY

6.1.2: Basic function call.

Complete the function definition to output the hours given minutes. Output for sample program:

```
3.5
```

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

void OutputMinutesAsHours(double origMinutes) {

   /* Your solution goes here  */
   cout << origMinutes/60;

}

int main() {
   double minutes;

   cin >> minutes;

   OutputMinutesAsHours(minutes); // Will be run with 210.0, 3600.0, and 0.0.
   cout << endl;

   return 0;
}
```

Run ✓ All tests passed

✓ Testing with input 210.0.

Your output

```
3.5
```

✓ Testing with input 3600.0.

Your output
```
60
```

✓ Testing with input 0.0.

Your output
```
0
```

Feedback?

CHALLENGE ACTIVITY

6.1.3: Function call with parameter: Printing formatted measurement.

Define a function PrintFeetInchShort, with int parameters numFeet and numInches, that prints using ' and " shorthand. End with a newline. Ex: PrintFeetInchShort(5, 8) prints:

```
5' 8"
```

Hint: Use \" to print a double quote.

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

/* Your solution goes here  */
void PrintFeetInchShort(int numFeet, int numInches){
   cout << numFeet << "' "<<numInches << "\""<< endl;

}

int main() {
   int userFeet;
   int userInches;

   cin >> userFeet;
   cin >> userInches;

   PrintFeetInchShort(userFeet, userInches);  // Will be run with (5, 8), then (4, 11)

   return 0;
}
```

Run ✓ All tests passed

✓ Testing with inputs: 5 8

Your output
```
5' 8"
```

✓ Testing with inputs: 4 11

Your output

```
4' 11"
```

Feedback?