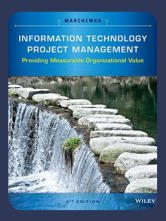
CH 07 QUIZ

Share

10 studiers recently ★ 5.0 (3 reviews)

Textbook solutions for this set

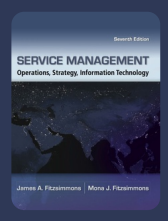


Information Technology Project Management: Providing Measurable Organizational Value

5th Edition • ISBN: 9781118898208

Jack T. Marchewka

346 solutions



Service Management: Operations, Strategy, and Information Technology

7th Edition • ISBN: 9780077475864

James Fitzsimmons, Mona Fitzsimmons

103 solutions

Search for a textbook or question ›

Terms in this set (63)

What prints?
 	
void fn(int, double, double&) { cout << "A" << endl; }
void fn(int, int, double&) { cout << "B" << endl; }
void fn(int, int, double) { cout << "C" << endl; }
void fn(int, int, int) { cout << "D" << endl; }
 	
int main()
{
auto n = 3.5;
fn(1, 2.5, n);
}
	A

What prints?
 	
void fn(int, double, double&) { cout << "A" << endl; }
void fn(int, int, double&) { cout << "B" << endl; }
void fn(int, int, double) { cout << "C" << endl; }
void fn(int, int, int) { cout << "D" << endl; }
 	
int main()
{
fn(2.5, 1.5, 2.5);
}
	C

What prints?
 	
void fn(int, double, double&) { cout << "A" << endl; }
void fn(int, int, double&) { cout << "B" << endl; }
void fn(int, int, double) { cout << "C" << endl; }
void fn(int, int, int) { cout << "D" << endl; }
 	
int main()
{
fn(1, 2, 3.5);
}
	C

What prints?

```
void fn(int, double, double&) { cout << "A" << endl; }
void fn(int, int, double&) { cout << "B" << endl; }
void fn(int, int, double) { cout << "C" << endl; }
void fn(int, int, int) { cout << "D" << endl; }

int main()
{
fn(2.5, 1.5, 7);
}
```

D

What prints?

```
void fn(int, double, double&) { cout << "A" << endl; }
void fn(int, int, double&) { cout << "B" << endl; }
void fn(int, int, double) { cout << "C" << endl; }
void fn(int, int, int) { cout << "D" << endl; }

int main()
{
fn(1, 2, 3, 4);
}
```

Syntax error: no candidates

What prints?

```
void fn(int, double, double&) { cout << "A" << endl; }
void fn(int, int, double&) { cout << "B" << endl; }
void fn(int, int, double) { cout << "C" << endl; }
void fn(int, int, int) { cout << "D" << endl; }

int main()
{
auto n = 3.5;
fn(1, 2, n);
}
```

Syntax error: ambiguous

What prints here?

```
auto a = 3, b = 3;
cout << (a != b ? "panda": "tiger") << endl;
```

tiger

What prints here?

```
auto a = 4, b = 3;
cout << (a == b ? "panda": a % 2 ? "stork": "tiger") << endl;
```

tiger

What prints here?

```
auto a = 3, b = 3;
cout << (a == b ? "panda": "tiger") << endl;
```

panda

What prints here?

```
auto a = 3, b = 3;
cout << (a != b ? "panda": a % 2 ? "stork": "tiger") << endl;
```

stork

What prints here?

```
auto a = 3, b = 3;
cout << a == b ? "panda" : "tiger" << endl;
```

Does not compile

Function overloading allows you to write several different functions that have the same name. Function overloading lets you call a single function in several different ways.	True False

Overloaded functions have the same name but different parameter types. Overloaded functions have the same name but different parameter names.	True False
In a while loop, (condition) is followed by a semicolon. A while loop is a hasty or unguarded loop.	False False

What prints here?
 	
auto a = 1;
switch (a)
{
case 1: cout << "1"; break;
case 2: cout << "2"; break;
default: cout << "3";
}
cout << endl;
	1

What prints here?
 	
auto a = 2;
switch (a)
{
case 1: cout << "1"; break;
case 2: cout << "2"; break;
default: cout << "3";
}
cout << endl;
	2

What prints here?
 	
auto a = '1';
switch (a)
{
case 1: cout << "1"; break;
case 2: cout << "2"; break;
default: cout << "3";
}
cout << endl;
	3

What prints here?
 	
auto a = 1;
switch (a)
{
case 1: cout << "1";
case 2: cout << "2";
}
cout << endl;
	12

What prints here?
 	
auto a = 1;
switch (a)
{
case 1: cout << "1";
case 2: cout << "2";
case 3:
}
cout << endl;
	Does not compile

What prints here?
 	
double a = 1;
switch (a)
{
case 1: cout << "1";
case 2: cout << "2";
}
cout << endl;
	Undefined behavior

What prints here?
 	
auto a = 'A';
switch (a)
{
case 64: cout << "?";
case 65: cout << "A";
case 66: cout << "B";
}
cout << endl;
	A
	But should be AB

The compiler determines which overloaded function to call by looking at the number, types and order of the arguments passed to the function.	True
Default arguments let you call a single function in several different ways. Default arguments allow you to write several different functions that have the same name.	True False
Default arguments may only be used with value parameters. Default arguments may only be used with reference parameters. Default arguments may be used with both value and reference parameters.	True False False
Default arguments appear only in the function prototype. Default arguments appear only in the function implementation.	True False
Fatal error messages should be printed to cerr. Fatal error messages should be printed to cout.	True False
Calling break() terminates a program immediately and passes an error code back to the operating system.	False

The compiler determines which overloaded function to call by looking at the type of value the function returns.	False
If str = "hello", then str.size() > -1.	False
Calling exit() terminates a program immediately and passes an error code back to the operating system.	True
A parameter with a default argument cannot appear before a parameter without a default argument.	True
A do-while loop is also called a hasty loop.	True

In a do-while loop, (condition) is followed by a semicolon.	True

To allow f() to change the argument passed here, the parameter str should be declared as:
 	
void f(. . . str);
int main()
{
string s = "hello";
f(s);
}
	string&

To allow f() to accept the argument passed here, the parameter str should be declared as:
 	
void f(. . . str);
int main()
{
f("hello");
}
	const string&

To allow f() to change the argument passed here, the parameter str should be declared as:
 	
void f(. . . str);
int main()
{
f("hello");
}
	It is not possible for f() to change the argument passed here.

A function where an argument is converted to match a parameter	best match
When more than one match is found for the proffered arguments.	ambiguity
A function where each argument is the same type as the corresponding parameter.	exact matches
A group of functions with the same name.	candidate set
A group of functions that have the same name and the correct number of parameters.	viable set
When no match is found for the proffered arguments	empty set

Examine the following variables and function calls
Match each item with the correct statement below.
int able = 3;
int baker = f1(able);
cout << able << baker << endl; // 64
 	
int charlie;
f2("hello", charlie);
cout << charlie << endl; // Hello Carl
	Returned value --> baker
	
	Output argument (parameter) --> Charlie
	
	Input argument (parameter) --> Hello
	
	Input/output argument (parameter) --> able

Which of these are not ways that functions may be overloaded?	different function name different return type different parameter names
Different functions that have the same name, but take different arguments, are said to be:	overloaded
You can call a single function in several different ways by giving the function _____:	default arguments

Given the overloaded functions prototypes and the variable definition below, which of the function calls will fail to compile?
 	
int f(int&);
int f(int);
int f(int, int);
int a = 7;
	f(a);

Given the overloaded functions prototypes and the variable definition below, which of the function calls will fail to compile?
 	
int f(int&);
int f(const int&);
int f(int, int);
int a = 7;
	None of these fail to compile

Assume that the input is 4 4 3 2 5. What will print?
 	
int i = 1;
int n;
cin >> n;
do
{
i++;
cin >> n;
}
while (n % 2);
cout << i << endl;
	2

Assume that the input is 5 5 4 3 5. What will print?
 	
int i = 1;
int n;
do
{
cin >> n;
i++;
}
while (n % 2);
cout << i << endl;
	4

int i = 1;
int n;
do
{
cin >> n;
i++;
}
while (n % 2);
cout << i << endl;
	lvalues

Examine this code. Which is the best prototype? int age; string name = read("Enter your name, age: ", age);	string read(const string&, int&)

What prints?
 	
string str = "Hello";
for (int i = str.size() - 1; i >= 0; i--)
cout << str.at(i);
	olleH

What prints?
 	
string str = "Hello";
for (size_t i = str.size() - 1; i >= 0; i--)
cout << str.at(i);
	Crashes when run

What prints?
 	
string str = "Hello";
for (auto i = 0, len = str.size(); i < len; i++)
cout << str.at(i);
	Does not compile

Which of these prototypes is the best one to use in this circumstance?
 	
int main()
{
string str{"To be or not to be."};
cout << "Most common letter is "
<< mostCommon(str) << endl;
}
	char mostCommon(const string&);

Which of these prototypes is the best one to use in this circumstance?
 	
int main()
{
string str{"TO BE OR NOT TO BE"};
properCase(str);
cout << str << endl;
}
	void properCase(string&);

Examine this code. Which is the best prototype?
 	
string s = "dog";
cout << upper(s) << endl; // DOG
cout << s << endl; // dog
	string upper(const string&)

Examine this code. Which is the best prototype?
 	
string s = "dog";
upper(s);
cout << s << endl; // DOG
	string upper(const string&)

Arguments passed to a function that has a non-constant reference parameter must be:	lvalues
A named constant, which can only be initialized once, is known as a _____.	non-modifiable lvalue
Arguments passed to a function that has a constant reference parameter must be:	either lvalues or rvalues are fine
The pattern of parameter types and order is called the function's:	signature

What prints here?
 	
int i = 5;
while (--i) cout << i;
cout << endl;
	4321

What prints here?
 	
int i = 5;
while (i--) cout << i;
cout << endl;
	43210

What prints here?
 	
int i = 5;
while (i) cout << --i;
cout << endl;
	43210

What prints here?
 	
int i = 5;
while (i) cout << i--;
cout << endl;
	54321

What prints here?
 	
int i = 5;
while (i); cout << i--;
cout << endl;
	Infinite loop