# Midterm 3 Study Guide

**Due** No due date **Points** 25 **Questions** 25 **Time Limit** 30 Minutes **Allowed Attempts** Unlimited

Take the Quiz Again

## Attempt History

| | Attempt | Time | Score |
|---|---|---|---|
| KEPT | Attempt 2 | 30 minutes | 22.5 out of 25 |
| LATEST | Attempt 3 | 26 minutes | 19.89 out of 25 |
| | Attempt 2 | 30 minutes | 22.5 out of 25 |
| | Attempt 1 | 27 minutes | 21 out of 25 |

ⓘ Correct answers are hidden.

Submitted Jul 12 at 3:41am

### Question 1

1 / 1 pts

Without `try` and `catch`, the `throw` statement terminates the running program.

- (●) True
- ( ) False

### Question 2

1 / 1 pts

Which of these are appropriate uses of the C++ `assert` facility?

- [x] Validate assumptions about your code
- [ ] Validate input received by your program
- [x] Validate the postcondition of a calculation
- [ ] Error conditions (such as file not found)
- [x] Debugging checks
- [x] Validate function arguments under the programmer's control

Partial

### Question 3

0.89 / 1 pts

*Match each item with the correct loop form below.*

| | |
|---|---|
| **Indefinite limit loop that reduces its input** | while (n != 0) { n /= 2; } |
| **Indefinite limit loop that uses successive approximations** | while(abs(g1 – g2) >= EPSILON |
| **Counter-controlled symmetric loop for producing a sequence of data** | for (int i=12; i <= 19; i++) {. . .} |
| **Indefinite data loop that uses raw input** | while(cin.get(ch)) {. . .} |
| **Counter-controlled asymmetric loop for processing characters** | for (int i=12; i <= 19; i++) {. . .} |
| **Iterator loop that may change its container** | for (auto& e : col) {. . .} |
| **Iterator loop that cannot change its container** | for (auto e : col) {. . .} |
| **Counter-controlled loop for processing substrings** | for (size_t i=4, slen=4, len=s.siz |
| **Indefinite data loop that uses formatted input** | while(cin >> n) {. . .} |

Incorrect

### Question 4

0 / 1 pts

What prints?

```
string s("hello");
try {
    if (s.size() > 2) throw s.size();
    if (islower(s.back())) throw s.back();
    if (s == "hello") throw string("hello");
    s.at(s.size()) = 'x';
    cout << "one\n";
```

```
}
catch (const int& e) { cout << "two\n"; }
catch (const string& e) { cout << "three\n"; }
catch (exception& e)  { cout << "four\n"; }
catch (...) { cout << "five\n"; }
```

- ○ one
- ○ four
- ◉ two
- ○ three
- ○ five
- ○ Undefined

Incorrect

## Question 5

0 / 1 pts

What prints?

```
string s("hello");
try {
    if (s.size() > 20) throw 42;
    if (isupper(s.back())) throw "goodbye";
    if (s == "Hello") throw string("hello");
    s[s.size()] = 'x';
    cout << "one\n";
}
catch (const int& e) { cout << "two\n"; }
catch (const string& e) { cout << "three\n"; }
catch (exception& e)  { cout << "four\n"; }
catch (...) { cout << "five\n"; }
```

- ○ three
- ○ one
- ○ four
- ○ two
- ○ Undefined
- ◉ five

## Question 6

1 / 1 pts

A `catch(…)` will catch any kind of thrown exception.

- ◉ True
- ○ False

## Question 7

1 / 1 pts

The C++11 standard library provides the function `stoi()` to convert a string to an integer. Which library is it found in?

- ○ iostream
- ○ cnvt
- ○ None of these
- ◉ string
- ○ cmath

## Question 8

1 / 1 pts

What happens when you execute the (erroneous) line:

```
bool ok = 2 + 2 == 5;
```

- An exception is thrown because 2 + 2 does not equal 5
- The program prints an error message and the program continues
- The program prints an error message and terminates
- The code does not compile, since 2 + 2 is not equal to 5
- Nothing happens. The variable ok is set to false.

## Question 9

1 / 1 pts

To use strings as a data stream source or sink, use the `<stringstream>` header

- True
- False

## Question 10

1 / 1 pts

What prints when this code runs?

```
enum class Coin
{
   PENNY = 1, NICKEL, DIME, QUARTER
};
cout << static_cast<int>(Coin::DIME) << endl;
```

- Does not compile; Missing semicolon at end of list of members.
- 10
- 3
- 2

## Question 11

1 / 1 pts

Examine the following code (which is legal). What changes are necessary to allow the statement `if (m1 == m2)` … to compile?

```
struct Money { int dollars{0}, cents{0}; } m1, m2;

bool equals(const Money& lhs, const Money& rhs)
{
    return lhs.cents == rhs.cents &&
        lhs.dollars == rhs.dollars;
}
```

- The name of `equals()` must be changed to `operator==`
- The function `equals()` must be declared inside the structure definition.
- The type `Money` needs to be a `class`
- This is not possible in C++.

## Question 12

1 / 1 pts

Assume that `v` contains `[1, 2, 3]`. The result of writing `cout << v.at(4);` throws a runtime exception.

- True
- False

## Question 13

1 / 1 pts

Structures data members must all be of the same type.

- True
- False

Incorrect

## Question 14

0 / 1 pts

What prints when this code runs?

```
enum class Coin
{
   PENNY, NICKEL, DIME, QUARTER
};
cout << static_cast<int>(Coin::PENNEY) << endl;
```

- (•) Does not compile; Cannot cast a `Coin` to an `int`.
- ( ) Does not compile; Missing semicolon at end of list of members.
- ( ) `0`
- ( ) `1`

Incorrect

## Question 15

0 / 1 pts

The declaration: `vector<int> v(10);` creates a `vector` object containing ten elements initialized to `0`.

- ( ) True
- (•) False

## Question 16

1 / 1 pts

`vector` is a structured library type.

- (•) True
- ( ) False

## Question 17

1 / 1 pts

The `pop_back` member function adds elements to the end of a `vector`.

- ( ) True
- (•) False

## Question 18

1 / 1 pts

Examine this version of the ***`swap()`*** function. How do you call it?

```
void swap(int * x, int & y)
{
    . . .
}
. . .
int a = 3, b = 7;
// What goes here ?
```

- ( ) `swap(a, b);`
- ( ) None of these
- ( ) `swap(&a, &b);`
- ( ) `swap(a, &b);`
- (•) `swap(&a, b);`

## Question 19

1 / 1 pts

The ***`reinterpret_cast`*** instruction produces a temporary value by converting its argument.

- ( ) True
- (•) False

Incorrect

## Question 20

0 / 1 pts

A forward reference can be used when you want to use a structure as a data member without first defining the entire structure.

- (●) True
- ( ) False

## Question 21

1 / 1 pts

Assume that ***ppi*** correctly points to ***pi***. Which line prints the address of ***ppi***?

```
int main()
{
    double pi = 3.14159;
    double *ppi;
    // code goes here
    // code goes here
}
```

- (●) `cout << &ppi;`
- ( ) `cout << *ppi;`
- ( ) None of these
- ( ) `cout << ppi;`
- ( ) `cout << &pi;`

## Question 22

1 / 1 pts

The elements of a C++ array created in a function are allocated on the stack.

- (●) True
- ( ) False

## Question 23

1 / 1 pts

All of these are legal C++ statements; which of them uses the C++ ***dereferencing operator***?

```
int a = 3, b = 4;
```

- ( ) None of these use the dereferencing operator.
- (●) `int x = *p;`
- ( ) `z *= a;`
- ( ) `int *p = &b;`
- ( ) `int y = a * b;`

## Question 24

1 / 1 pts

Which returns the last pixel on the first row of this image?

```
Pixel *p;   // address of pixel data
int w, h;   // width and height of image
```

- ( ) None of these are correct
- ( ) `p + w - 1`
- (●) `p[w - 1]`
- ( ) `p[w] - 1`
- ( ) `*p[w - 1]`

## Question 25

1 / 1 pts

Examine the following code. What is stored in ***a*** after it runs.

```
int f(int * p, int x)
{
    *p = x * 2;
    return x / 2;
}
. . .
int a = 3, b, c;
c = f(&b, a);
```

- ( ) 6

- 3
- 2
- Does not compile
- 1