Numbers

Practice Quiz • 10 min

Congratulations! You passed!

TO PASS 80% or higher

Keep Learning

GRADE

100%

Numbers

TOTAL POINTS 5

1. How many decimal digits could be stored in a signed 32-bit integer variable (also known as **int** in C++ and Java)?

1 / 1 point

9

Correct

Great! We could store any value from -2^{31} to $2^{31}-1$ and so any nine-digit decimal number, because $10^9 < 2^{31}-1$.

2. How many decimal digits could be stored in a signed 64-bit integer variable (also known as **long long** in C++ and **long** in Java)?

1 / 1 point

18

Correct

Well done! We could store any value from -2^{63} to $2^{63}-1$ and $2^{63}-1$ is a bit larger than $9 \cdot 10^{18}$, so 18 decimal digits do fit while 19 don't.

3. Check all fragments of code, where the overflow **does** happen. The code is given in C++, consider **int** to be a 32-bit signed integer, and **long long** — a 64-bit signed integer.

1 / 1 point

- [ ]

```
cout << (long long)10000 * 1000 * 10000 << '\n';
```

- [x]

```
int a = 100;
int b = 100000000;
cout << a * b << '\n';
```

Correct

The product is 10^{10}, and this is way more than 2 billion.

The product is $10^{[illegible]}$, and this is way more than 2 billion.

- [x]

```cpp
int n = 10000;
int m = 1000;
int res = 0;
for (int i = 1; i <= n; ++i) {
  res += i;
}
cout << res * m << '\n';
```

Correct

The sum of integers from 1 to N is about $N^2/2$, and so the final value is close to $5 \cdot 10^{10}$, which is too large for an int.

- [x]

```cpp
long long n = 100 * 1000;
long long m = 1000 * 1000;
long long res = 0;

for (int i = 0; i < n; ++i) {
  for (int j = 0; j < m; ++j) {
    res += 1000 * 1000;
  }
}

cout << res * 1000 << '\n';
```

Correct

The total value would be 10^{20}, and that is too large even for a long long — the maximal value for it is about $9 \cdot 10^{18}$.

- [ ]

```cpp
int n = 100000;
int m = 100;
int res = 0;
for (int i = 1; i <= n; ++i) {
  res += n / i;
}
cout << res * m << '\n';
```

4. How many decimal digits of **precision** could be stored in a **double** variable?

1 / 1 point

- ( ) 12
- ( ) 18
- ( ) 7
- (•) 15

Correct

Yes! There are 52 precision bits, and the first one is free, so 53 bits in total, and 2^{53} is just about $9 \cdot 10^{15}$.

5. Imagine that you should output a floating point number as the answer to some problem, and the statement says that the absolute or relative difference to the correct answer should be no more than 10^{-3}. Check those answer/output pairs, where the output would be accepted as correct, according to this rule.

1 / 1 point

- [x] 1000000.0 and 999013.0

> ✓ **Correct**
> The absolute difference is about thousand, but relative to 1000000 it's less than 0.001 — so the output would be accepted, as it suffices if only one of the differences is small enough.

- [ ] 5000.0 and 4982.76
- [x] 1.0 and 0.9991

> ✓ **Correct**
> The difference is less than 0.001, and it's both the absolute and the relative difference, because in the latter we divide on 1.

- [x] 0.0001 and 0.0009812

> ✓ **Correct**
> Both numbers aren't greater than 0.001 and are positive, so their absolute difference is surely smaller. Their relative difference is about 0.019 — but that's no problem, as it suffices if only one of them is small enough.