# Vectors, Casting, and Templates

**Lesson #2** - 09/11/2025

TCHS Programming Team

# Setup

Create a codeforces account at

https://codeforces.com

Once you have an account join the group by using the following url:

Also open the compiler to write code:

https://ide.usaco.guide

# casting

We can use variables with others of the same type. This means that, for example, we cannot compare `int` and `double` directly. To do this, we need to use **casting**.

```
double decimal_num = 4.69;

int number1 = decimal_num;

// what does this output?
cout << number1 << endl;
```

What is actually happening is the following:

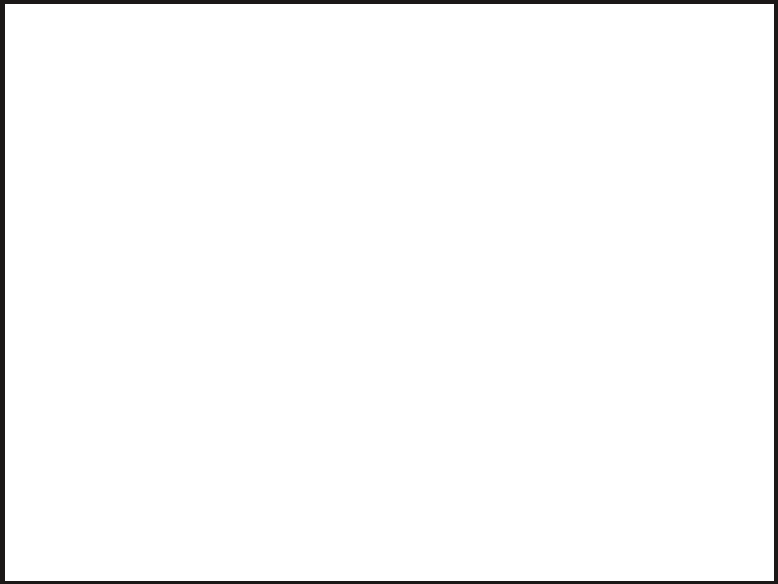
```
double decimal_num = 4.69;

int number1 = int(decimal_num);

// what does this output?
cout << number1 << endl;
```

# casting - Continued

```
int big_num = 500;

// 1e9 (or for any x -> 1ex for that matter) is
// actually a double
if(big_num < 1e9){
}

// so this if statement gets converted to:
if(double(big_num) < 1e9){
}

// since this is not desirable we can manually cast
// the double:
if(big_num < (int)1e9){
}
```

# Vectors

# Vectors - Concept



# Vectors - usage

Initializing vectors can be done like the following:

```
// where T is a data type
vector<T> my_list; // starts out empty - contains no elements)
vector<T> my_list(n); // starts with n-elements - they are all considered empty
vector<T> my_list(n,default_val); // starts with n-elements, all equal to default_val

vector<int> num_list;
vector<string> string_list;
vector<double> double_list; // etc...

// starts out with 5 elements, however each of them are considered empty
vector<int> empty_list(5);

// starts out with 5 elements, and all of them are equal to pi
vector<double> another_list(5,3.14567);

// T can even be another vector!
vector<vector<int>> num_list_2d(20, vector<int>(40, -5) );
// here num_list_2d is a list of 20 vectors, each of
// which has 40 elements all equal to -5
```

# Vectors - Looping through

```
vector<int> l = {1,2,3,4,5,6,7};

for(int i = 0;i < l.size();i++){
  // can access the current value by using l[i],
  // for example:
  cout << l[i] << " ";
}
cout << endl; // prints the end line

// if the index isn't needed then a
// for-each for loop can be used
for(int x : l){
  cout << x << " ";
}
cout << endl;


// for loop using iterators
for(auto it = l.begin(); it != l.end();it++){
  // value can be accessed at *it:
  cout << *it << " ";
}
cout << endl;
```

# Vectors - Looping through (ii)

For 2d vectors similar syntax can be used:

```
int n = 100;
int m = 200;

vector<vector<int>> dp(n, vector<int>(m, 0));

// this is for example how you can print a 2d list:
for(int i = 0;i < n;i++){
  for(int j = 0;j < n;j++){
    cout << dp[i][j] << " ";
  }
  cout << endl;
}
```

# Vectors - Methods

Vectors support lots of methods:

| method | comment |
|---|---|
| `push_back(T value)` | add a value to the list. |
| `size()` | Returns the number of elements in the vector. |
| `front()` | Returns the first element of the vector |
| `back()` | Returns the last element of the vector |
| `erase(iterator)` | Erases value at some index* |
| `empty()` | Returns true if empty, otherwise returns false. |

```
vector<int> l = {10,20,30,50,20};
l.push_back(-10);
l.push_back(l.begin() + 2);
cout << l.size() << endl;
for(int x : l)
  cout << x << " ";
```

```
output:
5
10 20 50 20 -10
```

# Programming Problems' structure / related topics

# Reading in a list / lists

A common pattern is reading in a list of numbers:

```
// --- first approach --- //
int n;
cin >> n; // read in number of elements

// list starts empty
vector<int> nums;

// read in elements
for(int i = 0;i < n;i++){
  int tmp; cin >> tmp;
  nums.push_back(tmp); // adds each element into the list
}

// --- second approach (recommended) --- //
int n;
cin >> n; // read in number of elements

// create a list with n elements
// NOTE: you must initialize the list with n-elements before reading them in,
// otherwise you will get an error!
vector<int> nums(n);

// read in elements
for(int i = 0;i < n;i++){
  cin >> nums[i];
}
```

# Test cases

Input for each test case is two numbers:

| Single test case | Multiple Test cases |
|---|---|
| `2 3 // test case #1` | `3 // number of test cases`<br>`2 3 // test case #1`<br>`6 7 // text case #2`<br>`4 1 // test case #3` |

Input for each test case is a list:

| Single test case | Multiple Test cases |
|---|---|
| `5`<br>`1 2 5 3 4` | `2 // number of test cases`<br>`5 // start of test case 1`<br>`1 2 5 3 4`<br>`3 // start of test case 2`<br>`1 2 3` |

# Templates

Some code can often be annoying to type over and over, so we can make templates to make writing solutions easier

# Template

```cpp
#include <bits/stdc++.h>
using namespace std;

// the list of shortcuts will likely grow over the year
using ll = long long;
using vi = vector<int>;

void solve(){
  // same as `long long long_num;`
  ll long_num = (ll)1e16;
  // same as `vector<int> list_of_nums;`
  vi list_of_nums;

  // both are the same:
  vector<vector<int>> list_2d(10, vector<int>(40, 30) );
  vector<vi> list_2d(10, vi(40, 30) );
}

int main(){
  int tc = 1;
  cin >> tc; // comment this out if its only one test case
  while(tc--){
    solve();
  }
}
```

# planning stuff

TODO: plan example code for each of the slides:

- casting first slide: have code to show off casting
- one / multiple test cases

# things to know when presenting

- first ask who have used lists, 2d lists, and

vectors in specific

- mention that my goals is to introduce topics, but that they are responsible for actually practicing

and researching what I cannot cover

- functions
- topics for next lesson:
- pairs
- sets
- time complexity

# Miscellaneous

- Practice competition
  - planning to have it this coming Tuesday
- feedback on slides
  - theme
- next meeting
  - sets / hash maps
  - functions / recursion