Substrings

To create a new string, initialized with only a portion of an existing string (called **a substring**), use the member function named `substr()` which takes two parameters:

- the index of the **first character** you want to select
- the desired **number of characters**.

Calling `str.substr(start, n)` creates **a new `string`** by extracting `n` characters from `str` starting at the index position specified by `start`. For example, if `str` contains the `string` `"hello, world"`, then the following code prints `"ell"`.

```
string str{"hello, world"};
cout << str.substr(1, 3) << endl;
```

The `string` begins at `0`, so the character at index `1` is the character `'e'`.

Be careful with the `substr()` function, when switching between Java and C++. In Java, the second parameter to its `substring()` method is the ending index; in C++, though, it is the number of characters in the returned substring. Forgetting this can lead to hard-to-find bugs (and crashes).

The second argument in `substr()` **optional**; if missing, `substr()` returns the substring that starts at the index and continues to the end. For instance,

```
cout << str.substr(7) << endl;
```

returns the `string` `"world"`. While this line

```
cout << str.substr(str.size() / 2) << endl;
```

uses `substr()` to print the second half of `str`, which includes the middle character if the size of `str` is odd:

*When using the `substr(start, end)` version of `substr()`, if `n` is supplied but fewer than `n` characters follow the starting position, `substr()` returns characters only up to the end of the original `string`, instead of causing a runtime error. If, however, `start` is beyond the length of the `string`, **you will** get an error. If `start` is equal to the length of the `string`, then `substr()` returns the empty string.*