CHAPTER

ONE

C++ LANGUAGE CONSTRUCTS: WEEK01

1.1 Fixed-size Arrays

1.2 Integer Data Types

Register overflow happens silently without warning:

```
int numbers[5] = {1000000001, 1000000002,
                  1000000003, 1000000004, 1000000005};
int sz = sizeof(numbers)/sizeof(int);  // assigns array size (=5).
int iTotal;             // 4 byte register (signed)
long unsigned luTotal;  // 4 bytes register (unsigned)
long long llTotal;      // 8 bytes register (signed)
for (int i = 0; i < sz; i++) {
  intTotal += numbers[i];
      luTotal += numbers[i];
      llTotal += numbers[i];
      cout << i << "-th partial sum is " << intTotal << "(int), "
    << luTotal << "(long unsigned), "
    << llTotal << "(long long)." << endl;
}
```

1.3 Float Data Types

```
int success = 4;
int total = 7;
cout << "Wrong proportion: " << 1.0*(success/total) << endl;
cout << "Correct proportion: " << (1.0*success/total) << endl;
// output exactly 6 decimal places:
cout << fixed << setprecision(6) << (1.0*success/total) << endl;
```

1.4 Text Data Types

```
void staircase(int n) {
  for (int i = 1; i <= n; i++) {
    string spaces(n - i,' ');
    string hashes(i, '#');
    cout << spaces << hashes << "\n";
  }
}
```

1.5 Vectors

Vectors are list-like objects that can hold objects of the same type, for example, `vector<int>` means a vector of `int` variables (4-byte integer numbers). See Initialize a vector in C++.

```
// initialize a vector with the given elements
vector<int> primes{2,3,5,7,11,13,17,19,23,29};
```

```
int vectorSumAsArray(vector<int> v) {
  int result = 0;
  for (int i = 0; i < v.size(); i++) {
    result += v[i];
  }
  return result;
}
```

If you want to access vector without using "array syntax" (`v[i]`), you can use method `v.at(i)` to access i-th element of a vector. In this case we have a square-sized vector of vectors and add up the elements on its diagonal:

```
int diagonalDifference(vector<vector<int>> vv) {
  int diag = 0;
  for (int i = 0; i < arr.size(); i++) {
    diag += arr.at(i).at(i);
  }
  return diag;
}
```

Here is another method to operate with vectors: create iterator and loop over it. For vectors it is not much different from the above method (accessing vector like an array). But for some data structures iterator is more flexible as it does not need to know the size of the data structure.

CHAPTER

TWO

ASSIGNMENT: WEEK01

1. Here is some stuff:

```
int numbers[5] = {1000000001, 1000000002,
                  1000000003, 1000000004, 1000000005};
int sz = sizeof(numbers)/sizeof(int);  // assigns array size (=5).
int iTotal;             // 4 byte register (signed)
```

2. Here is another stuff:

```
int numbers[5] = {1000000001, 1000000002,
                  1000000003, 1000000004, 1000000005};
int sz = sizeof(numbers)/sizeof(int);  // assigns array size (=5).
int iTotal;             // 4 byte register (signed)
```