4. (15%) Write a function: `void average_word_lenght(string & sentence, float & result)` that calculates the average length of all words in the string `sentence`.

```
#include <string>
#include <iostream>
using namespace std;

void average_word_lenght(string & sentence, float & result) {
    float sum = 0;
    int count = 1;
    for (int i = 0; i < sentence.length(); i++)
    {
        if (sentence[i] == ' ') {
            count++;
        }
        else {
            sum++;
        }
    }
    result = sum / count;
}

int main()
{
    string sentence = "";
    float result;
    average_word_lenght(sentence, result);
    cout << result << endl;
    return 0;
}
```

5. (15%) Write a program that asks user for a positive integer side length. If they enter an illegal value, they must be prompted to enter a good one until they do. It then displays, using asterisks, a filled diamond of the given side length. For example, if the side length is 4, the program should display:

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

```cpp
int main()
{
   cout << "Enter the length of the diamond side: ";
   int side;
   cin >> side;
   // Calculate the max width of the diamond
   int max_width = (side * 2) - 1;
   // Calculate half of that to place spaces
   int half_width = (max_width) / 2;
   int dots = 1;
   // Print top half of diamond
   for (int i = 0; i < side; i++)
   {
      // Print spaces to line up triangle
      for (int j = 0; j < half_width; j++)
      {
         cout << " ";
      }
      // Print dots
      for (int j = 0; j < dots; j++)
      {
         cout <<"*";
      }
      cout << endl;
      dots = dots + 2;
      half_width--;
   }
   // Reset variables for bottom half
   dots = max_width - 2;
   half_width = 1;
   // Print bottom half of diamond
   for (int i = 0; i < side; i++)
   {
      // Print spaces to line up triangle
      for (int j = 0; j < half_width; j++)
      {
         cout << " ";
      }
      // Print dots
      for (int j = 0; j < dots; j++)
      {
         cout <<"*";
      }
      cout << endl;
      dots = dots - 2;
      half_width++;
   }
   return 0;
}
```

Variable and Constant Definitions

Type *Name* *Initial value*

```
int cans_per_pack = 6;

const double CAN_VOLUME = 0.335;
```

Mathematical Operations

```
#include <cmath>
```

`pow(x, y)`	Raising to a power x^y
`sqrt(x)`	Square root $\sqrt{x}$
`log10(x)`	Decimal log $\log_{10}(x)$
`abs(x)`	Absolute value $\lvert x \rvert$
`sin(x)` `cos(x)` `tan(x)`	Sine, cosine, tangent of x (x in radians)

Selected Operators and Their Precedence

(See Appendix B for the complete list.)

`[]`	Array element access
`++ -- !`	Increment, decrement, Boolean *not*
`* / %`	Multiplication, division, remainder
`+ -`	Addition, subtraction
`< <= > >=`	Comparisons
`== !=`	Equal, not equal
`&&`	Boolean *and*
`||`	Boolean *or*
`=`	Assignment

Loop Statements

Condition

```
while (balance < TARGET)
{
   year++;
   balance = balance * (1 + rate / 100);
}
```

Executed while condition is true

Initialization *Condition* *Update*

```
for (int i = 0; i < 10; i++)
{
   cout << i << endl;
}
```

Loop body executed at least once

```
do
{
   cout << "Enter a positive integer: ";
   cin >> input;
}
while (input <= 0);
```

Conditional Statement

Condition

```
if (floor >= 13)
{
   actual_floor = floor - 1;
}
else if (floor >= 0)
{
   actual_floor = floor;
}
else
{
   cout << "Floor negative" << endl;
}
```

Executed when condition is true

Second condition (optional)

Executed when all conditions are false (optional)

String Operations

```
#include <string>
string s = "Hello";
int n = s.length(); // 5
string t = s.substr(1, 3); // "ell"
string c = s.substr(2, 1); // "l"
char ch = s[2]; // 'l'
for (int i = 0; i < s.length(); i++)
{
   string c = s.substr(i, 1);
   or char ch = s[i];
   Process c or ch
}
```

Function Definitions

Return type *Parameter type and name*

```
double cube_volume(double side_length)
{
   double vol = side_length * side_length * side_length;
   return vol;
}
```

Exits function and returns result.

Reference parameter

```
void deposit(double& balance, double amount)
{
   balance = balance + amount;
}
```

Modifies supplied argument

Arrays

Element type *Length*

```
int numbers[5];
int squares[] = { 0, 1, 4, 9, 16 };
int magic_square[4][4] =
{
   { 16, 3, 2, 13 },
   { 5, 10, 11, 8 },
   { 9, 6, 7, 12 },
   { 4, 15, 14, 1 }
};

for (int i = 0; i < size; i++)
{
   Process numbers[i]
}
```