```cpp
/*  Program: Dice Simulator
    Author: Tom Stutler
    Last Date Modified: 3/19/15

    The intent of this program is to repeatedly prompt the user for a number of times they would like to toss a a pair of dice (between 1-100,000),
    then simulate the tosses, tally the amount of times each sum (2-12) is tossed, calculate the probability of each sum being tossed,
    then display the results to the user in a formatted table.
*/
//#define NDEBUG
#include <cassert>
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

//Define global constants for array lengths.
const int MAX_ROLLS = 100001, POSSIBLE_SUMS = 11;

void rolldie (int dieParam[], int rollsParam);
///This function takes in an empty integer array and a positive integer,
///then simulates rolling a die and storing the outcome maxParam times.

void findsum (int die1Param[], int die2Param[], int sumsParam[], int rollsParam);
///This function takes in two arrays with simulated dice rolls, calculates the
///sum of the two dice, then stores the sums to a new array.

void tosscount (int sumsParam[], int countParam[], int rollsParam);
///This function takes an array of sums and counts how many times each possible
///sum was rolled. It then stores the counts to a new array.

void display (int countParam[], int rollsParam);
///This function takes an array with the tally of how many times each possible sum was
///rolled and the amount of times the user choose to roll the dice then displays the
///number of rolls, the possible sums, the tally of each sum, and the probability of each sum.

int main()
{
    //Define loop check variable.
    char repeat;

    //Loop to repeat until user decides to stop.
    do {
        //Define variables.
        int qtyRolls, dieRolls_1[MAX_ROLLS], dieRolls_2[MAX_ROLLS], sumRolls[MAX_ROLLS], sumCount[POSSIBLE_SUMS] = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0};

        //Seed rand().
        srand(time(NULL));

        //Loop to ensure the user enters a valid input.
        do {
            //Prompt user for the desired number of tosses (between 1-100,000) and stores to variable.
            cout << "Enter number of tosses (1-100000): ";
            cin >> qtyRolls;
        } while ((qtyRolls < 1) || (qtyRolls > 100000));

        assert((qtyRolls >= 1) && (qtyRolls <= 100000));
        //Simulate rolling the dice.
        rolldie(dieRolls_1, qtyRolls);
        rolldie(dieRolls_2, qtyRolls);

        //Calculate the sums of the two dice for each roll.
        findsum(dieRolls_1, dieRolls_2, sumRolls, qtyRolls);

        //Tally the counts of how many times each sum was rolled.
        tosscount(sumRolls, sumCount, qtyRolls);

        //Display the results to the user.
        display(sumCount, qtyRolls);

        //Prompt the user if they want to repeat.
        cout << "Do another simulation? (y or n): ";
        cin >> repeat;

    } while ((repeat == 'y') || (repeat == 'Y'));
}

void rolldie (int dieParam[], int rollsParam)
{
    int i;

    for (i=0; i<rollsParam; i++) {
        dieParam[i] = rand() % 6 + 1;
        assert((1 <= dieParam[i]) && (dieParam[i] <= 6));
    }
}

void findsum (int die1Param[], int die2Param[], int sumsParam[], int rollsParam)
{
    for (int i=0; i<rollsParam; i++) {
        sumsParam[i] = die1Param[i] + die2Param[i];
        assert((2 <= sumsParam[i]) && (sumsParam[i] <= 12));
    }
}

void tosscount (int sumsParam[], int countParam[], int rollsParam)
{
    for (int i=0; i < rollsParam; i++) {
        countParam[sumsParam[i]-2]++;
    }
}

void display (int countParam[], int rollsParam)
{
    int i;

    cout << "Total number of tosses = " << rollsParam << endl;
    cout << "\tToss\tCount\tProbability\n";

    for (i=0; i < POSSIBLE_SUMS; i++) {
        cout << "\t" << i+2 << "\t" << countParam[i] << "\t" << (static_cast<float>(countParam[i])/static_cast<float>(rollsParam))*100 << endl;
    }
}
```