```
#include <iostream>
#include <vector>
#include <cmath>
#include <cstdlib>


// Fonction pour générer des données selon une équation mathématique
std::vector<double> createData(int count, float dx)
{
  std::vector<double> data(count); // Allocation d'un vecteur de taille 'count'

  for (int i = 0; i < count; i++) {
    float x = dx * i;
    data[i] = 5 * exp(-x) * cos(10*x) + 0.5 * (rand()) / RAND_MAX;
  }

  return data;
}


void average(std::vector<double> data, int nb, float& result)
{
  result = 0;

  for(int k = 0; k < nb; k++) {
    result += data[k];
  }
}

int main()
{
  std::vector<double> y1 = createData(100, 0.05);
  std::vector<float> y2(y1.size());

  int n = 6;

  for (int i = n/2; i < y1.size() - n/2; i++) {
    float avg = 0;
    average(std::vector<double>(y1.begin() + i, y1.begin() + i + n), n, avg);
    y2.push_back(avg);
  }

  for (const auto& value : y1) {
    std::cout << value << std::endl;
  }
}

// ************************************************************
/*
* Sinon en utilisant des itérateurs
*/

// Fonction pour calculer la moyenne d'une plage d'éléments avec des itérateurs
void average(std::vector<float>::const_iterator begin, std::vector<float>::const_iterator end,
float& result) {
  result = std::accumulate(begin, end, 0.0f) / std::distance(begin, end);
}

int main() {
  std::vector<float> y1 = createData(100, 0.05f); // Génération des données
  int n = 6; // Taille de la fenêtre pour le calcul de la moyenne glissante
  std::vector<float> y2(100 - n); // Allocation de la taille appropriée pour le stockage des
moyennes

  // Boucle pour appliquer une moyenne glissante sur les données
  for (int k = n / 2; k < 100 - n / 2; k++) {
```

```
    average(y1.begin() + k - n / 2, y1.begin() + k - n / 2 + n, y2[k - n / 2]);
  }

  // Affichage des valeurs générées
  for (const auto& value : y1) {
    std::cout << value << std::endl;
  }

  return 0;
}
```