/\* Составить и вывести на экран новый массив с номерами элементов исходного массива,

\* которые равны заданному значению.

\* заданное значение вводится с клавиатуры

\*/

**Рассмотрим различные способы реализации в порядке убывания приоритета (от наиболее предпочтительного к наименее предпочтительному**)

***Способ 1: использование статических массивов***

(Размеры массивов задаются константой. Формируемый массив index определяем такой же размерности как и исходный, выделение памяти под массивы происходит на этапе компиляции программы)

#include <iostream>

using namespace std;

int main() {

const int n = 100;

int array[n]; // source array

int index[n]; // resulting array

int size;

cin >> size; // real size of source array

  for (int i = 0; i < size; ++i) { // filling the array

    cin >> array[i];

  }

  int value;

  cout << "Value: ";

  cin >> value; // set value

  int k =0;

  for (int i = 0; i < size; i++){

    if (array[i] == value){

      index[k++] = i;

    }

  }

  cout << "Index(es) of value: ";

  for (int j =0; j <k; j++){

    cout << index[j] << " ";

  }

  return 0;

}

***Способ 2: использование vector для формируемого массива***

(размерность формируемого массива index определяется на этапе выполнения программы)

#include <iostream>

#include <vector>

using namespace std;

int main() {

  const int n = 100;

int array[n]; // source array

int size; // size of array

  cout << "Size: ";

  cin >> size;

  cout << "Array: ";

  for (int i = 0; i < size; i++){ // filling the array

    cin >> array[i];

  }

  int value;

  cout << "Value: ";

  cin >> value;

  vector <int> index; // vector indexes of value

// vector <int> index {}; //

  for (int i = 0; i < size; i++){

    if (array[i] == value){

      index.push\_back(i);

    }

  }

  cout << "Index(es) of value: ";

  for (auto x: index){

    cout << x << " ";

  }

  return 0;

}

Когда переменная-вектор выходит из области видимости, то она автоматически освобождает память, которую занимала.

***Способ 2а: оба массива с помощью vector (плюс здесь показана проверка сформированного массива на пустоту)***

#include <iostream>

#include <vector>

using namespace std;

int main() {

int size; // size of array

cout << "Size: ";

cin >> size;

vector <int> array; // source vector

cout << "Array: ";

for (int i = 0; i < size; ++i){ // filling the array

int j;

cin >> j;

array.push\_back(j);

}

int value;

cout << "Value: ";

cin >> value;

vector <int> index; // vector indexes of value

for (int i = 0; i < size; ++i){

if (array[i] == value){

index.push\_back(i);

}

}

if (index.empty()) { // check if the array is empty

  cout << "no such numbers";

}

else {

cout << "Index(es) of value: ";

  for (auto x: index){

  cout << x << " ";

}

}

return 0;

}

***Способ 3: использование динамических массивов (доступ к элементам массива с помощью указателя)***

(память под оба массива запрашивается во время выполнения программы, причем обработка исходного массива array на сравнение его элементов с заданным значением производится дважды: сначала для определения количества таких элементов, чтобы выделить память под формируемый массив index, а затем чтобы сформировать index)

#include <iostream>

using namespace std;

int main() {

int size; // size of array

 cout << "Size: ";

 cin >> size;

 int \*array = new int [size]; // memory allocation for array

 cout << "Array: ";

 for (int i = 0; i < size; i++){ // filling the array

    cin >> \*(array +i);

  }

  int value;

  cout << "Value: ";

  cin >> value;

  int k =0;

  for (int i = 0; i < size; i++){

    if (\*(array +i) == value){

      k++;

    }

  }

int \*index = new int[k];

int x = 0;

for (int i = 0; i < size; i++) {

if (\*(array +i) == value){

      \*(index+x++) = i;

    }

  }

  cout << "Index(es) of value: ";

for (int x = 0; x < k; x++)

    cout << \*(index + x) << " ";

delete [] index;

delete [] array;

return 0;

}

new и delete образуют связанную пару, размер массива для освобождения памяти соответствует размеру созданного соответствующим new

***Способ 3а: использование динамических массивов (доступ к элементам массива с помощью индекса)***

#include <iostream>

using namespace std;

int main() {

  int size; // size of array

  cout << "Size: ";

  cin >> size;

  int \*array = new int [size];

  cout << "Array: ";

  for (int i = 0; i < size; ++i){ // filling the array

    cin >> array[i];

  }

  int value;

  cout << "Value: ";

  cin >> value;

  int k =0;

  for (int i = 0; i < size; ++i){

    if (array[i] == value){

      k++;

    }

  }

int \*index = new int[k];

int x = 0;

for (int i = 0; i < size; i++) {

if (array[i] == value){

      index[x++] = i;

    }

  }

  cout << "Index(es) of value: ";

for (int x = 0; x < k; x++)

    cout << index[x] << " ";

delete [] index;

delete [] array;

return 0;

}

***Способ 4: (НЕ характерно для С++, память под оба массива запрашивается во время выполнения программы, работает не во всех средах, например, в Microsoft Visual Studio такой способ не будет работать)***

#include <iostream>

using namespace std;

int main() {

 int size; // size of array

cout << "Size: ";

  cin >> size;

  int array[size]; // !!!

  cout << "Array: ";

  for (int i = 0; i < size; ++i){ // filling the array

    cin >> array[i];

  }

  int value;

  cout << "Value: ";

  cin >> value;

int k = 0;

for (int i = 0; i < size; ++i){

    if (array[i] == value){

      k++;

    }

  }

int index [k]; //!!!

int x = 0;

  for (int i = 0; i < size; ++i){

    if (array[i] == value){

      index [x++] = i;

    }

  }

  cout << "Index(es) of value: ";

  for (int x = 0; x < k; x++){

    cout << index [x] << " ";

  }

  return 0;

}