Ravesli Ravesli

- Уроки по C++
- OpenGL
- SFML
- Qt5
- RegExp
- Ассемблер
- Купить .PDF

Урок №91. Цикл foreach

Юрий |

- Уроки C++

|

Обновл. 23 Авг 2020 |

73395

| 22

На этом уроке мы рассмотрим использование цикла foreach в языке C++.

Оглавление:

Цикл foreach

На **уроке №76** мы рассматривали примеры использования **цикла for** для осуществления итерации по каждому элементу массива. Например:

```
1  #include <iostream>
2
3  int main()
4  {
5      const int numStudents = 7;
6      int scores[numStudents] = { 45, 87, 55, 68, 80, 90, 58 };
```

```
    int maxScore = 0; // отслеживаем наивысший балл
    for (int student = 0; student < numStudents; ++student)
        if (scores[student] > maxScore)
            maxScore = scores[student];

    std::cout << "The best score was " << maxScore << '\n';

    return 0;
}
```

В то время как циклы for предоставляют удобный и гибкий способ итерации по массиву, в них так же легко можно запутаться и наделать «ошибок неучтенных единиц».

Поэтому в C++11 добавили новый тип цикла — **foreach** (или *«цикл, основанный на диапазоне»*), который предоставляет более простой и безопасный способ итерации по массиву (или по любой другой структуре типа списка).

Синтаксис цикла foreach следующий:

```
for (объявление_элемента : массив)
   стейтмент;
```

Выполняется итерация по каждому элементу массива, присваивая значение текущего элемента массива переменной, объявленной как **элемент** (`объявление_элемента`). В целях улучшения производительности объявляемый элемент должен быть того же типа, что и элементы массива, иначе произойдет **неявное преобразование**. Рассмотрим простой пример использования цикла foreach для вывода всех элементов массива `math`:

```
#include <iostream>

int main()
{
    int math[] = { 0, 1, 4, 5, 7, 8, 10, 12, 15, 17, 30, 41};
    for (int number : math) // итерация по массиву math
       std::cout << number << ' '; // получаем доступ к элементу массива в этой итерац

    return 0;
}
```

Результат выполнения программы:

```
0 1 4 5 7 8 10 12 15 17 30 41
```

Рассмотрим детально, как это всё работает. При выполнении цикла foreach переменной `number` присваивается значение первого элемента (т.е. значение `0`). Дальше программа выполняет стейтмент вывода значения переменной `number`, т.е. нуля. Затем цикл for выполняется снова, и значением переменной `number` уже является `1` (второй элемент массива). Вывод значения `number` выполняется снова. Цикл продолжает свое выполнение до тех пор, пока в массиве не останется непройденных элементов. В конце выполнения программа возвращает `0` обратно в операционную систему с помощью оператора return.

Обратите внимание, переменная `number` не является индексом массива. Ей просто присваивается значение элемента массива в текущей итерации цикла.

Цикл foreach и ключевое слово auto

Поскольку объявляемый элемент цикла foreach должен быть того же типа, что и элементы массива, то это идеальный случай для использования **ключевого слова auto**, когда мы позволяем C++ вычислить тип данных элементов массива вместо нас. Например:

```
#include <iostream>

int main()
{
    int math[] = { 0, 1, 4, 5, 7, 8, 10, 12, 15, 17, 30, 41 };
    for (auto number : math) // тип number определяется автоматически исходя из типа э
        std::cout << number << ' ';

    return 0;
}
```

Цикл foreach и ссылки

В примерах, приведенных выше, объявляемый элемент всегда является переменной:

```
    int array[7] = { 10, 8, 6, 5, 4, 3, 1 };
    for (auto element: array) // element будет копией текущего элемента массива
        std::cout << element << ' ';
```

То есть каждый обработанный элемент массива копируется в переменную `element`. А это копирование может оказаться затратным, в большинстве случаев мы можем просто ссылаться на исходный элемент с помощью **ссылки**:

```
    int array[7] = { 10, 8, 6, 5, 4, 3, 1 };
    for (auto &element: array) // символ амперсанда делает element ссылкой на текущий э
        std::cout << element << ' ';
```

В примере, приведенном выше, в качестве объявляемого элемента цикла foreach используется ссылка на текущий элемент массива, при этом копирования этого элемента не происходит. Но, с указанием обычной ссылки, любые изменения элемента будут влиять на сам массив, что не всегда может быть желательно.

Конечно же, хорошей идеей будет сделать объявляемый элемент константным, тогда вы сможете его использовать в режиме «*только для чтения*»:

```
    int array[7] = { 10, 8, 6, 5, 4, 3, 1 };
    for (const auto &element: array) // element - это константная ссылка на текущий эле
```

```
3         std::cout << element << ' ';
```

Правило: Используйте обычные ссылки или константные ссылки в качестве объявляемого элемента в цикле foreach (в целях улучшения производительности).

Еще один пример

Вот пример первой программы из начала этого урока, но уже с использованием цикла foreach:

```
#include <iostream>

int main()
{
    const int numStudents = 7;
    int scores[numStudents] = { 45, 87, 55, 68, 80, 90, 58};
    int maxScore = 0; // отслеживаем индекс наибольшего score (значения)
    for (const auto &score: scores) // итерация по массиву, присваиваем каждое значени
        if (score > maxScore)
            maxScore = score;

    std::cout << "The best score was " << maxScore << '\n';

    return 0;
}
```

Обратите внимание, здесь нам уже не нужно вручную прописывать индексацию массива. Мы можем получить доступ к каждому элементу массива непосредственно через переменную `score`.

Цикл foreach и не массивы

Циклы foreach работают не только с **фиксированными массивами**, но также и со многими другими структурами типа списка такими, как векторы (например, std::vector), связанные списки, деревья. Не беспокойтесь, если вы не знаете, что это такое (мы всё это рассмотрим чуть позже). Просто помните, что циклы foreach обеспечивают гибкий и удобный способ итерации не только по массивам:

```
#include <vector>
#include <iostream>

int main()
{
    std::vector<int> math = { 0, 1, 4, 5, 7, 8, 10, 12, 15, 17, 30, 41}; // обратите в
    for (const auto &number : math)
        std::cout << number << ' ';

    return 0;
}
```

Цикл foreach не работает с указателями на массив

Для итерации по массиву, цикл foreach должен знать длину массива. Поскольку массивы, которые **распадаются в указатель**, не знают своей длины, то циклы foreach с ними работать не могут!

```
#include <iostream>

int sumArray(int array[]) // array - это указатель
{
    int sum = 0;
    for (const auto &number : array) // ошибка компиляции, размер массива неизвестен
        sum += number;

    return sum;
}

int main()
{
     int array[7] = { 10, 8, 6, 5, 4, 3, 1 };
     std::cout << sumArray(array); // array распадается в указатель здесь
     return 0;
}
```

По этой же причине циклы foreach не работают с **динамическими массивами**.

Могу ли я получить индекс текущего элемента?

Циклы foreach не предоставляют прямой способ получения индекса текущего элемента массива. Это связано с тем, что большинство структур, с которыми могут использоваться циклы foreach (например, связанные списки), напрямую не индексируются!

Заключение

Циклы foreach обеспечивают лучший синтаксис для итерации по массиву, когда нам нужно получить доступ ко всем элементам массива в последовательном порядке. Эти циклы предпочтительнее использовать вместо стандартных циклов for в случаях, когда они могут использоваться. Для предотвращения создания копий каждого элемента в качестве объявляемого элемента следует использовать ссылку.

Тест

Это должно быть легко!

Объявите фиксированный массив со следующими именами: `Sasha`, `Ivan`, `John`, `Orlando`, `Leonardo`, `Nina`, `Anton` и `Molly`. Попросите пользователя ввести имя. Используйте цикл foreach для проверки того, не находится ли имя, введенное пользователем, уже в массиве.

Пример результата выполнения программы:

```
Enter a name: Sasha
Sasha was found.

Enter a name: Maruna
Maruna was not found.
```

Подсказка: Используйте **std::string** в качестве типа массива.

Ответ

```
#include <iostream>
#include <string>

int main()
{
    const std::string names[] = { "Sasha", "Ivan", "John", "Orlando", "Leonardo", "Nin

    std::cout << "Enter a name: ";
    std::string username;
    std::cin >> username;

    bool found(false);
    for (const auto &name : names)
        if (name == username)
        {
            found = true;
            break;
        }

    if (found)
        std::cout << username << " was found.\n";
    else
        std::cout << username << " was not found.\n";

    return 0;
}
```

Оценить статью:

(**227** оценок, среднее: **4,92** из 5)

Урок №90. Оператор доступа к членам через указатель

Урок №92. Указатели типа void

Комментариев: 22

1.  *Артурка*:
   29 октября 2020 в 00:23

   Разве не правильние говорить не foreach цикл, а range based for? Так как в STL хедере <algorithm> есть функция for_each(InputIt first, InputIt last, UnaryFunction f) и далее может возникнуть путаница в терминах при изучении.

   Ответить

   1.  *Юрий*:
      29 октября 2020 в 17:48

      Хороший вопрос.

      Дело в том, что есть уже устоявшиеся «народные» термины, «цикл foreach» относится к таковым. Да, «foreach» не является ключевым словом, и при использовании в коде мы нигде не указываем «foreach». Но это «народное название», которое позволяет сразу понять, о чем идет речь.

      range based for — это тоже полноценное название цикла foreach, которое можно использовать и которое тоже в своем названии передает суть данного цикла. Можно использовать как первый вариант, так и второй. В статье указан первый вариант в качестве основного.

      То, что есть функция for_each() — это ничего (по большей мере) не значит. Повторюсь, ключевого слова "foreach" нет, следовательно и спутать конкретную функцию for_each() с "народным" названием цикла трудно, т.к. в коде цикл foreach объявляется с помощью ключевого слова for. А в устной речи мы всегда конкретизируем, говорим ли мы о цикле, или о функции.

      Ответить

2.  *Andrey*:
   5 октября 2020 в 22:33

```
#include <iostream>

int sumArray(int array[], const int &size) // array - это указатель
{
    int sum = 0;
```

```
    int val[7];
    for(int i = 0; i < size; ++i)
    {
        val[i]=array[i];
    }
    for (const auto &number : val)
        sum += number;

    return sum;
}

int main()
{
    const int size = 7;
     int array[7] = { 10, 8, 6, 5, 4, 3, 1 };
     std::cout << sumArray(array,size); // array распадается в указатель здесь
     return 0;
}
```

Можно провести итерацию , если только переданный массив скопировать в другой массив в функции

Ответить

3.  *Vlad*:
7 сентября 2020 в 23:18

```
#include <iostream>

int main()
{
    using namespace std;

    cout << "Enter Name: ";
    string inputName; cin >> inputName;

    const string names[]{ "Sasha", "Ivan", "John", "Orlando", "Leonardo", "Nina",
    {
        bool isExist(false);
        for (const auto& name : names)
            if (name == inputName)
            {
                isExist = true;
                break;
            }

        cout << inputName << " was " << (isExist ? "" : " not") << " found.\n";
    }
```

```
22
23      return 0;
24  }
```

Ответить

4.  *Onium*:
[7 июля 2020 в 22:42](#)

```
1   #include <iostream>
2   #include <string>
3
4   int main()
5   {
6       std::string math[] = {"Sasha", "Ivan", "John",
7           "Orlando", "Leonardo", "Nina", "Anton", "Molly"};
8
9       std::cout << "Enter a name: ";
10      std::string name;
11      std::cin >> name;
12
13
14      for (const auto& name1 : math)
15      {
16          if (name==name1)
17          {
18              std::cout << name << " was found" << '\n';
19              break;
20          }
21          else if(name!=name1)
22          {
23              std::cout << name << " was not found" << '\n';
24              break;
25          }
26
27      }
28
29
30
31
32      return 0;
33  }
```

Ответить

1.  *VEX*:
[21 июля 2020 в 14:03](#)

Неправильно, у тебя сравнивается ввод с элементов массива "Sasha"и если ввод совпадает с "Sasha", то он выводит, что нашёл, и завершает цикл, а в противном случае он выводит, что не нашёл и выходит из цикла, а остальные элементы не обрабатываются

Ответить

5.  *Кетчуп*:
1 июля 2020 в 17:14

Допустим что в цикле foreach есть массив numbers и элемент number.
Если вспомнить что под капотом ссылок указатели и ссылка это разыменованный указатель, то легко понять что индекс элемента будет равен &number — numbers, потому что numbers + n = & (numbers[n]).
(https://ravesli.com/urok-83-adresnaya-arifmetika-i-indeksatsiya-massiva/)

Ответить

1.  *Viktor*:
21 сентября 2020 в 21:47

Не очень с тобой соглашусь по двум пунктам.
1) как я читал на многих форумах не регламентируется, то как ссылка должна быть реализована внутри. К сожалению, пока не нашел способ почитать сам стандарт без покупки за большую сумму.
2) массив не обязан хранить элементы, совпадающие с индексом. Кроме этого, элементы массива могут быть вообще не int типа

Ответить

6.  *Bampi*:
18 июня 2020 в 02:41

Куда более интересный вопрос: откуда for(auto& ind : arr) знает размер массива?

Если правильно все понял, то при объявлении масства, например:

```
int arr[10];
```

имеет представление int[10] и из

```
template <typename T, size_t N>
inline constexpr size_t
arrLen(const T (&arr) [N]) {
    return N;
}
```

Откуда for знает, что массив состоит из 10 элементов. Или как-то по другому?

Ответить

7.  *Павел*:
[1 июня 2020 в 17:22](#)

Блин думал над таким же решением как в ответе, но подумал нафиг плодить сущности))), так что мой вариант решения)))

```
#include <iostream>
#include <string>

using namespace std;

int main()
{
    setlocale(LC_ALL, "ru");

    string name[] = { "Sasha", "Ivan", "John", "Orlando", "Leonardo", "Nina", "An
    string input;

    cout << "Введите имя: ";
    cin >> input;

    for (const auto& score : name) // итерация по массиву, присваиваем каждое зна
        {
            if (input == score)
            {
                cout << "Такое имя есть в списке\n";
                system("pause");
                return 0;
            }
        }

    cout << "Нет такого имени\n";

   system("pause");
   return 0;
}
```

Ответить

8.  *Константин*:
[22 ноября 2019 в 11:14](#)

И правда не сложно!

```
#include <iostream>
```

```
#include <string>

int main()
{
    std::string nameList[] = { "Sasha", "Ivan", "John", "Orlando", "Leonardo", "N
    std::cout << "Enter a name: ";
    std::string xname;
    std::cin >> xname;
    bool eqNames = false;

    for (auto& name : nameList)
    {
        eqNames = (xname == name);
        if (eqNames)
            break;
    }
    if (eqNames)
        std::cout << xname << " was found\n";
    else
        std::cout << xname << " was not found";

    return 0;
}
```

Ответить

9.  *Евгений*:
[14 ноября 2019 в 14:45](#)

Решил потренироваться с динамической памятью, чтобы потом указать размер фиксированного массива. Но я так понял у меня это не выйдет?
И мне нужно вместо этого создать динамический массив с таким размером?

```
int *ptr = new int { 6553600 };
    cout << ptr;
    std::string array[*ptr] {"Sasha", "Ivan", "John", "Orlando", "Leonardo", "Nina
```

Ответить

10.  *Nikita*:
[27 августа 2019 в 22:19](#)

Почему у всех в комментариях есть переменная bool? Я что-то пропустил?

```
#include "pch.h"
#include <iostream>
#include <string>
```

```
4
5  using namespace std;
6
7      int main()
8      {
9          string names[8]{ "Sasha","Ivan","John","Orlando","Leonardo","Nina","Anton
10         cout << "Введите имя: ";
11         string name;
12         cin >> name;
13         for (auto &test : names)
14         {
15             if (test == name)
16             {
17                 cout << "Name " << name << " is found." << endl;
18                 return 0;
19             }
20         }
21         cout << "Name " << name << " not found." << endl;
22     }
```

Ответить

1.  *Алексей*:
   10 ноября 2019 в 19:11

   Написать можно как угодно, это работает.
   Но Вы не написали, что берете const auto &array, в серьезных программах может быть беда.

   Второе — не написано return 0 в конце main.

   Сам снова тупанул написав в цикл проверку без прерывания.

   Ответить

11.  *Алексей*:
    20 августа 2019 в 17:46

    Тест простой, правда тупанул насчет проверки, зачем-то влепил в цикл.
    Хотел сделать проверку на ввод — не работает. Ту же пустую строку.

```
1  #include <iostream>
2  #include <string>
3  #include <conio.h>
4
5  int main()
6  {
7      std::string names[]{ "Sasha", "Ivan", "John", "Orlando", "Leonardo", "Nina",
8
```

```
    std::string name;
    bool check = false;
    bool ch;

    do {
        std::cout << "Enter a name:";
        std::cin >> name;
        if (name == "\n") { std::cin.clear(); std::cin.ignore(32767, '\n'); ch = 
        else { ch = false; }
    } while (check);

    system("CLS");
    for (const auto &checkname : names)
    {
        if (name == checkname)
        {
            check = true; break;
        }
    }

    if (check)
    {
        std::cout << name << " was found." << '\n';
    }
    else
    {
        std::cout << "Name wasn't found!" << '\n';
    }

    return 0;
}
```

Ответить

1.  *Павел*:
   1 июня 2020 в 17:32

   Не понял

```
do {
    std::cout << "Enter a name:";
    std::cin >> name;

    if (name == "\n")
    {
        std::cin.clear();
        std::cin.ignore(32767, '\n');
        ch = true;
```

```
        system("CLS");
    }

    else { ch = false; }

} while (check);
```

Поправьте если я что то не так понял, вот это "if (name == "\n")" означает условие начала с новой строки, или если человек нажал ENTER? Потом вроде всё понятно зачищаем всё что можно в том числе и данные с экрана. Затем новый непонятный стэймент "while (check)" это что?

Объясните плиз, а то не понимаю(((

Ответить

12.  *Анастасия*:
29 июня 2019 в 18:34

Я использую getline, чтобы записать введённую строку в переменную. Вопрос:
почему если я перед этим очищаю cin с помощью

```
std::cin.ignore(1000, '\n');
```

то у меня после ввода имени идёт ещё одна, пустая строка и потом, соответственно, пишет, что такого имени нет. Без cin.ignore(1000, '\n'); всё находит нормально и пустой строки нет.
Вот мой код:

```
#include <iostream>
#include <clocale>  //для вывода текста на кириллице
#include <string>   //для std::string

using std::cout; using std::cin; using std::endl;

int main()
{
    setlocale(LC_CTYPE, "rus"); //для вывода текста на кириллице

    // просим пользователя ввести имя:
    cout << "Введите имя, используя латинский алфавит: ";
    //cin.ignore(1000, '\n'); // если раскомментировать эту строку, будет ещё одна
    std::string usersName{};
    std::getline(cin, usersName);

    // создаём массив имён из задания:
    std::string namesArray[] = { "Sasha", "Ivan", "John", "Orlando", "Leonardo",

    // ищем заданное пользователем имя в массиве, если нашли, прерываем цикл:
    bool found = false;
```

```
    for (const auto& name : namesArray)
    {
        if (name == usersName)
        {
            cout << "Такое имя в списке есть!" << endl;
            found = true;
            break;
        }
    }

    // если по итогу не нашли, выводим сообщение об этом:
    if (found==false)
        cout << "Такого имени в списке нет!" << endl;

    return 0;
}
```

Ответить

1.  *Михаил*:
   26 июля 2019 в 20:26

   Ты этим не очищаешь cin, а просишь ввести строку, которая после ввода будет просто отброшена. И поэтому первая строка, что ты вводишь отбрасывается, а вторая уже записывается в переменную.

   Ответить

   1.  *Анастасия*:
      28 июля 2019 в 17:22

      перечитала объяснение про cin.ignore, разобралась, спасибо!

      Ответить

13.  *Вячеслав*:
    21 апреля 2019 в 21:11

    у меня так получилось

```
#include "pch.h"
#include <iostream>
#include <string>

int main()
{
    using namespace std;
```

```
    setlocale(0, "");

    const string names[] = { "Sasha", "Ivan", "John", "Orlando", "Leonardo", "Nin
    string name1;
    cout << " Введите имя ";
    cin >> name1;

    bool flag(false);
    for (const auto &name : names)
    {
        if (name == name1)
            flag = true;
        break;
    }
    if (flag)
        cout << name1 << " найдено. \n";
    else
        cout << name1 << " не найдено.\n";
    return 0;
}
```

Ответить

14.  *Павел*:
[29 января 2019 в 17:38](#)

У меня решение в тексте выдает сразу "Enter a name: was not found."

Ответить

15.  *Игорь*:
[23 сентября 2018 в 15:44](#)

```
#include <iostream>

void search();
int main()
{
    search();
    return 0;
}
void search()
{
    std::string userName;
    std::string stringMasName[] = { "Sasha", "Ivan", "Orlando", "Leonardo",
                                    "Nina", "Anton", "John", "Molly" };
    std::cout<<"Enter a name: ";
```

```
15      std::cin>>userName;
16      bool flag = false;
17
18      for(const auto &name:stringMasName)
19      {
20          if(name == userName)
21              flag = true;
22
23          if(flag)
24              break;
25      }
26
27      std::cout<<userName<<" was ";
28      std::cout<<( (flag)? "found." :"not found.");
```

Ответить

Добавить комментарий

Ваш E-mail не будет опубликован. Обязательные поля помечены *

Имя *

Email *

Комментарий

Сохранить моё Имя и E-mail. Видеть комментарии, отправленные на модерацию

Получать уведомления о новых комментариях по электронной почте. Вы можете подписаться без комментирования.

Отправить комментарий

TELEGRAM КАНАЛ

ПАБЛИК

ТОП СТАТЬИ

- Словарь программиста. Сленг, который должен знать каждый кодер

- Урок №1. Введение в программирование
- 70+ бесплатных ресурсов для изучения программирования
- Урок №1: Введение в создание игры «SameGame» на C++/MFC
- Урок №4. Установка IDE (Интегрированной Среды Разработки)