ПОВТОРНЫЕ ОПЕРАТОРЫ

Мамаражабов Жахонгир Шерзод ўғли

Студент кафедры компьютерных наук и технологий программирования факультета информационных технологий Термезского государственного университета

jahongirmamarajabov780@gmail.com

Аннотация

Еще одним мощным механизмом управления выполнением программы являются операторы повторения. Оператор повторения используется для многократного повторения операторов (тела повторения) в определенной части программы относительно истинного значения выражения, известного как «условие повторения». В этой статье рассматриваются операторы повторения и то, как писать программы с использованием этих операторов, тестировать и выполнять различные части кода на основе результатов тестирования.

Ключевые слова: Повторяющиеся операторы, для повторяющегося оператора, повторяющегося оператора, программа, сеть, переменные, программа ветвления.

Abstract

Another powerful mechanism for controlling program execution is repetition statements. The repetition operator is used to repeatedly repeat statements (repetition body) in a certain part of the program relative to the true value of the expression, known as the "repeat condition". This article discusses the repetition operators and how to write programs using these operators, test, and execute different parts of the code based on the test results.

Keywords: Repeating statements, for repeating statement, repeating statement, program, network, variables, branching program.

Цикл имеет свои точки входа и выхода, но может не иметь точки выхода. В этом случае повторение называется бесконечным повторением. Для бесконечного цикла условие продолжения цикла всегда истинно.

Проверки условий цикла могут быть проверены перед выполнением операторов в теле цикла (циклы for, while) или после того, как операторы в теле

цикла были выполнены один раз (do-while).

Операторы повторения могут быть вложенными.

***for* оператор повторения: [1(119-127), 3(37-41), 4(55-56)]** *for* синтаксис оператора повторения следующий:

for (<выражение >; < выражение >;< выражение >*) ;*</ выражение </ выражение </ выражение

Этот оператор начинается с выполнения выражения < выражение >. Затем начинаются шаги повторения. На каждом шаге выполняется <выражение>, если результат отличен от 0 или истина, выполняется тело итерации - и в конце выполняется <выражение>. Если <выражение> равно 0 (false), итерация останавливается и управление переходит от оператора итерации к следующему оператору. Следует отметить, что выражение <выражение> может быть комбинацией нескольких выражений, разделенных запятыми, и в этом случае значением последнего выражения является условие повторения. Телом итерации может быть один оператор, включая пустой оператор, или блок операторов.

Например, рассмотрим задачу вычисления суммы целых чисел от 10 до 20.

```
#include
int main()
{int S=0;
for (int i=10; i<=20; i++)
S+=i;
cout<<’’S=’’ <<S;
return 0;}
```

Оператор повторения в программе начинает свою работу с присвоения параметру повторения I (счетчику повторений) начального значения 10, и после каждого шага повторения (итерации) его значение увеличивается на единицу (за счет выполнения третьего оператора в скобки). На каждом шаге итерации выполняется оператор в теле итерации, т. е. к переменной Sum добавляется значение 1. Когда значение счетчика итераций i равно 21, возникает условие итерации «i<=20» (значение 0), и итерация заканчивается. В результате управление переходит от оператора repeat к следующему оператору cout, а сумма выводится на экран.

Заключенные в скобки выражения операторов повторения можно интерпретировать в соответствии с приведенным выше примером:

<выражение> — служит для инициализации переменной, которая действует как счетчик итераций и считается только один раз в начале итерации. Объявление переменной может встречаться в выражении, и эта переменная допустима в теле оператора итерации и не "видима" вне оператора итерации (для компилятора C++ Builder);

<выражение> — это логическое выражение, указывающее, следует ли выполнять итерацию, если условие истинно, то итерация продолжается, в противном случае — нет. Если это выражение пустое, условие всегда истинно;

<выражение> - обычно служит для увеличения (уменьшения) значения счетчика повторений, либо может иметь другие действия, влияющие на условие повторения.

Оператор повторения не может содержать выражения в круглых скобках, но синтаксис не допускает отсутствия ';'. Таким образом, простой оператор повторения будет выглядеть так:

for(;;) cout <<” Бесконечное повторение …’’;

Если при итерации необходимо синхронно изменить значение нескольких переменных, этого можно добиться, написав необходимые операторы с ',' в выражениях <выражение> и <выражение>:

```
for(int i=10 , j=2 ; i<=20 ; i++ , j=i+10)
{
…
}
```

На каждом шаге оператора итерации соответственно изменяются значения переменных j и I.

Оператор for может не иметь тела итерации. Например, если необходимо «приостановить» выполнение программы на определенный период времени, этого можно добиться, выполнив повторение без выполнения какой-либо дополнительной работы:

```
#include
int main()
{
int delay;
…
for (delay=5000; delay>0; delay--);// пустой оператор
…
```

```
return 0;
}
```

Сумму приведенных выше чисел от 10 до 20 можно вычислить с помощью оператора итерации пустого тела (пустого оператора):

```
…
for (int i=10; i<=20; summa+=i++) ;
…
```

Использование блока операторов в качестве тела оператора повторения можно показать на примере факторного вычисления:

```
#include
int main()
{
int a;
unsigned long fact=1;
cout<<’’ введите целое число:_’’;
cin>>a;
if ((a>=0)&&(a<33))
{
for (int i=1; i<=a; i++) fact*=i;
cout<<a<<<fact<<’\n’;
}
return 0;
}
```

Программа выполняется, когда пользователь вводит число от 0 до 33, потому что 34! Значение не помещается в пробелы, зарезервированные для unsigned long.

Foydalanilgan adabiyotlar

1. Brinkley, D. L., & Schell, R. R. (1995). Concepts and terminology for computer security. Information Security: An integrated collection of essays, 40-97.

2. William G.J. Halfond, Jeremy Viegas and Alessandro Orso, “A Classification of SQL Injection Attacks and Countermeasures,” College of Computing Georgia Institute of Technology IEEE, 2006.

3. OWASP: Top 10 Security Threats 2013. Retrieved online: https://www.owasp.org/index.php/Top_10_2013-A1-Injection (2013)

4. Wilander, J., & Kamkar, M. (2003, February). A Comparison of Publicly Available Tools for Dynamic Buffer Overflow Prevention. In NDSS (Vol. 3, pp. 149-162)

5. US-CERT. Vulnerability notes database. Retrieved online: www.kb.cert.org/vuls.