آموزش C++ - درس ۷- حلقه ها
ساختارهای Do، While، For

به سری آموزشهای C++ سایت About که توسط سایت Persian-Designers ترجمه شده است خوش آمدید. این درس سه نوع حلقه که در برنامه های C++ استفاده می شود را پوشش می دهد. حلقه ها برای اجرای قطعه ای از کد به تعداد مشخص ساخته می شوند. همچنین حلقه ها می توانند برای اجرای تکراری قطعه ای از کد تا تغییر حالت یک شرط منطقی، ساخته شوند. برای مثال، یک حلقه ممکن است تا تغییر حالت یک شرط از false به true ادامه یابند، یا از true به false. در این حالت، قطعه ای از کد که اجرا شده باید شرط بررسی شده را در درون حلقه به روز در آورد تا حلقه در جایی پایان پذیرد. اگر شرط بررسی شده به هر طریق در داخل حلقه تغییر داده نشود، حلقه هیچگاه پایان نمی پذیرد. این کار باعث خطایی خواهد شد که آن را به عنوان حلقه بی نهایت می شناسند. (در برنامه نویسی ویندوز، بخصوص برنامه نویسی بازیها وضعیت کمی فرق می کند که در بخشهای آینده که مربوط به برنامه نویسی ویندوز خواهد بود در مورد آن بحث خواهد شد- مترجم)

حلقه while
حلقه while برای اجرای قطعه ای از کد در طی زمانی است که مقدار شرط true باشد. اگر مقدار شرط false از ابتدا false باشد حلقه اصلا اجرا نخواهد شد. حالت کلی آن به این شکل است:

```
while (tested condition is satisfied)
{
   block of code
}
```

در اینجا یک مثال ساده در مورد استفاده از while را ببینید. این برنامه از ۱ تا ۱۰۰ را می شمارد.

```
#include <iostream>
using namespace std;

int main()
{
   int count = 1;

   while (count <= 100)
   {
      cout << count << endl;
      count += 1; //Shorthand for count = count + 1
   }

   return 0;
}
```

در اینجا یک مثال واقعی تر از استفاده از while را ببینید. این برنامه حداقل تعداد بیتهای مورد نیاز برای ذخیره یک عدد صحیح مثبت را تعیین می کند.بزرگترین عدد بدون علامت که می تواند در N بیت ذخیره شود $2^N - 1$ است.

```
#include <iostream>
using namespace std;

int main()
{
   int bitsRequired = 1; //the power of 2
   int largest = 1; //largest number that can be stored
   int powerOf2 = 2;
   int number;

   cout << "Enter a positive integer: ";
   cin >> number;
   while (number > largest)
   {
      bitsRequired += 1; //Shorthand for bitsRequired = bitsRequired + 1
      powerOf2 = powerOf2 * 2;
      largest = powerOf2 - 1;
   }

   cout << "To store " << number << " requires ";
   cout << bitsRequired << " bits" << endl;
```

```
    return 0;
}
```

حلقه do

حلقه do قطعه ای از کد را تا زمانی که مقدار شرط درست باشد اجرا می کند. تفاوت بین حلقه do و حلقه while در این است که حلقه while شرط را در بالای حلقه بررسی می کند، حلقه do شرط را در پایین حلقه اش بررسی می کند. همانطوری که اشاره شد، اگر شرط بررسی شده false باشد حلقه while از روی بلاک کد مربوطه پرش می کند. از آنجایی که شرط در پایین حلقه do بررسی می شود، بلاک کد مربوط به آن همیشه حداقل یک مرتبه اجرا می شود. حالت کلی حلقه های do بدین شکل است:

```
do {
    block of code
} while (condition is satisfied)
```

در اینجا مثالی از چگونگی استفاده از حلقه do را می بینید. برنامه ای را که مشاهده می کنید یک بازی است که اجازه می دهد کاربر یک عدد بین ۱ تا ۱۰۰ را حدس بزند. حلقه do مناسب است زیرا ما می دانیم که برنده بازی همواره حداقل به یک بار اجرا نیاز دارد.

```
#include <iostream>
using namespace std;

int main()
{

    int number = 44;
    int guess;

    cout << "Guess a number between 1 and 100" << endl;
    do {
        cout << "Enter your guess: ";
        cin >> guess;

        if (guess > number) {
            cout << "Too high" << endl;
        }
        if (guess < number) {
            cout << "Too low" << endl;
        }
    } while (guess != number);

    cout << "You win. The answer is " << number << endl;

    return 0;
}
```

حلقه for

سومین ساختار حلقه در C++ حلقه for است. حلقه for می تواند قطعه ای از کد را برای تعداد مشخصی تکرار اجرا نماید. حالت کلی آن بدین شکل است:

```
for (initializations;test conditions;actions)
{
    block of code
}
```

ساده ترین راه برای یادگیری حلقه for مطالعه چند مثال است. اولین مثال، در اینجا یک حلقه for از ۱ تا ۱۰ را می شمارد.

```
for (int count = 1; count <= 10; count++)
{
    cout << count << endl;
}
```

شرط حلقه ممکن است مرتبط با متغیرهای در حال مقداردهی شده و به روز شده نباشد. در اینجا یک حلقه است که تا زمانی که کاربر حلقه را پایان ندهد، به عمل شمارش ادامه می دهد.

```
for (count = 1; response != 'N'; count++)
{
   cout << count << endl;
   cout << "Continue (Y/N): ";
   cin >> response;
}
```

شرطهای پیچیده تر نیز وجود دارد. فرض کنید کاربر آخرین مثال هرگز "N" را وارد نکند، اما وقتی که حلقه به ۱۰۰ رسید باید خاتمه یابد.

```
for (count = 1; (response != 'N') && (count <= 100); count++)
{
   cout << count << endl;
   cout << "Continue (Y/N): ";
   cin >> response;
}
```

همچنین این امکان را داریم که چندین مقداردهی اولیه و چندین عمل را داشته باشیم. این حلقه یک شمارنده را از ۰ شروع می کند و شمارنده دیگر را از ۱۰۰ و نقطه میانی بین آنها را پیدا می کند.

```
for (i = 0, j = 100; j != i; i++, j--)
{
   cout << i << " " << j << endl;
}
cout << i << " " << j << endl;
```

Initializations are optional. For instance, suppose we need to count from a user specified number to 100. The first semicolon is still required as a place keeper.

مقداردهی اولیه اختیاری است. برای مثال، فرض کنید ما باید از یک عدد تعیین شده توسط کاربر تا ۱۰۰ را بشماریم. نقطه ویرگول ابتدا هنوز مورد نیاز است تا جای آن حفظ شود.

```
cout << "Enter a number to start the count: ";
cin >> count;
for ( ; count < 100 ; count++)
{
   cout << count << endl;
}
```

اعمال نیز اختیاری هستند. در اینجا مثالی است که مکررا یک عدد را تا زمانی که کاربر حلقه را قطع کند، نمایش می دهد.

```
for (number = 5; response != 'Y';) {
   cout << number;
   cout << "Had Enough (Y/N) " << endl;
   cin >> response;
}
```

تمرین: سعی کنید برنامه هایی بنویسید که از ۱۰ تا ۱ را با استفاده از حلقه های while، do، for بشمارد.

ترجمه: محمد صافدل
منبع: http://cplus.about.com