دستورات کنترلی

این بخش دستورات کنترلی که راهی برای کنترل اجرای برنامه ها در C++ هستند را می پوشاند. این دستورات شامل دستورات شرطی if-else و switch و حلقه های تکرار while، do-while و for هستند.

ترتیب اجرای برنامه ها در C++ به صورت top-down است. بدین معنی که اجرا از ابتدای تابع main() آغاز می شود دستور به دستور ادامه پیدا می کند تا به انتهای main() برسد. دستورات کنترلی موجود در C++ اجازه تغیي ترتیب اجرای برنامه را می دهند.

دستورات شرطی که از ساختارهای تصمیم گیری هستند دستوراتی هستند که در صورت برقرار بودن شرطی دستوری را اجرا می کنند. دو دستور شرط در C++ وجود دارد: if و switch.

دستور if-else

معمول ترین نوع ساختار تصمیم گیری if است. یک دستور if می گوید "اگر شرطی برقرار است کد معینی را انجام بده". دستور if-else به دو شکل می تواند باشد: همراه با else یا بدون آن.

```
if(expression)
   {
    statement
   }
```

یا

```
if(expression)
   {
    statement
   }
else
   {
    statement
   }
```

بدنبال کلمه if یک پرانتز می آید که عبارت شرط در آن قرار دارد و یک مقدار بولین false یا true تولید می کند (در C به صورت صفر و غیر صفر ارزیابی می شود). با درست و غلط بودن عبارت شرطی مسیر اجرا تعیین می شود. اگر درست باشد دستور بعد از if اجرا می شود. اگر else وجود داشته باشد در صورت برقرار نبودن شرط دستور بعد از else اجرا می شود.

مثال. شرط ساده.

```
if (age==18)
    {
     cout <<"Yahoo…. You old enough to drive!\n";
    }
```

مثال. شرط کامل.

```
#include <iostream>

int main()
{
    cout << "Enter your age: ";
    int age;
    cin >> age;

    if (age < 20)
        cout << "You are still young!\n";
    else
        cout << "You are not so young anymore.\n";
}
```

ممکن است دستور if-else بدون محصور شدن در آکولاد بیاید. آکولاد محدوده بلاک کد را معین می کند. اگر بلاک کد تنها شامل یک دستور باشد کامپایلر فرض می کند اولین جمله بعد از if (یا بعد از else) بلاک کد است که باید اجرا شود.

مثال. در شرط زیر وقتی شرط برقرار است فقط ;y++ اجرا می شود. خط دوم جز دستور if نیست و صرفنظر از درست بودن شرط همواره اجرا می شود.

```
if (x == 7)
y++;
z++;
```

شرط موردنیاز در دستور if توسط عملگرهای رابطه ای >، <، =>، =<، == و =! ساخته می شود. دو یا چند شرط می توانند توسط عملگرهای منطقی &&(AND)، || (OR) و ! (NOT) با هم ترکیب شوند.

مثال. چند نمونه عبارت شرطی.

```
if (x>6)
if (x !=6)
if (5<j<10)
if (j== 5 && I == 6)
if (j<4 || j>10)
```

بخاطر داشته باشید که یک علامت مساوی عملگر انتساب است و دوتا مساوی عملگر هم ارزی به معنی "آیا مساوی است با". a==b درست است اگر a برابر با b باشد. a=b مقدار b را به a نسبت می دهد. یکی از بیشترین خطاهائی که مبتدیان می کنند استفاده از علامت = به جای = = است. اگر تک مساوی را استفاده کنید شرط if همیشه درست است.

شرط های تودرتو

یک امکان دیگر دستور if دستورات if تودرتو است یعنی یک if درون دیگری.

مثال. اگر Num مثبت باشد عبارت positive اگر صفر باشد عبارت zero و درغیراینصورت اگر منفی باشد عبارت negative نمایش داده می شود.

```
if (Num > 0)
   cout << " is positive" << endl << endl;
else if (Num == 0)
   cout << " is zero" << endl << endl;
else
   cout << " is negative" << endl << endl;
```

دستور switch

دستورات if و if-else برای یک یا دو حالت انتخاب مناسب هستند اما برای انتخاب های بیشتر به جای یک سری از if های تودرتوانتخاب دیگری هم وجود دارد. دستور switch بر اساس مقدار یک متغیر یا عبارت صحیح روی کدی که می خواهید سوئیچ می کند. شکل کلی آن به صورت زیر است:

```
switch(selector) {
    case integral-value1 : statement; break;
    case integral-value2 : statement; break;
    case integral-value3 : statement; break;
    case integral-value4 : statement; break;
    (...)
    default: statement;
}
```

selector باید یک متغیرصحیح یا عبارتی باشد که یک عدد صحیح تولید می کند. این مقدار با مقادیر صحیح بعد از case ها مقایسه می شود اگر برابر با یکی از آنها بود دستور مربوط به آن اجرا می شود. اگر برابر با هیچکدام از مقادیر نبود دستور بعد از default اجرا می شود.

دستور switch دارای یک بلاک از کد است. آکولاد درست بعد از switch شروع و بعد از آخرین دستور آن تمام می شود.

مثال.

```
int choice;
switch(choice) {
    case 1:
       cout<<"you chose 1\n";
       break;
    case 2:
       cout<<"you chose 2\n";
       break;
    case 3:
       cout<<"you chose 3\n";
       break;
    default:
       cout<<"you made an invalid choice \n";
    }
```

در انتهای هر case یک دستور break وجود دارد که باعث می شود اجرا به انتهای بلاک switch منتقل شود. اگر break حذف شود اجرا ادامه پیدا می کند و کلیه case های بعدی اجرا می شود تا وقتی که با یک دستور break مواجه شود.

مثال.

```
char Choice;
cout << "Enter the letter for your choice: ";
cin >> Choice;
switch(Choice)
   {
   case 'A':
   case 'a':
      DoOptionA();
      break;
   case 'B':
   case 'b':
      DoOptionB();
      break;
   case 'C':
   case 'c':
      DoOptionC();
   }
```

دستور switch راهی برای ساختن شرط های تودرتو است اما به selector احتیاج دارد که بر اساس مقدار آن تصمیم گیری کند. وقتی مفید است که می خواهید مقدار یک متغیر یا عبارت صحیح با یک سری ثابت مقایسه و کد مربوط به آن اجرا شود.

نکته. در هر case تنها از یک مقدار ثابت می توان استفاده کرد.
نکته. عبارت رشته ای به عنوان selector نمی تواند بکار رود. برای یک عبارت رشته ای باید از شرط های تودرتو استفاده کرد.
نکته. می توان دستورات switch تودرتو نیز داشت.
نکته. دستورات درون هر case می تواند بیشتر از یکی باشد و به آکولاد برای محصور کردن آنها نیازی نیست.

حلقه های for

یک حلقه (loop) دستوری است تا زمان برآورده شدن شرطی دستورات دیگر را تکرار می کند. دستور داخل حلقه می تواند بلاکی از کد، حلقه یا هر دستور دیگری باشد.

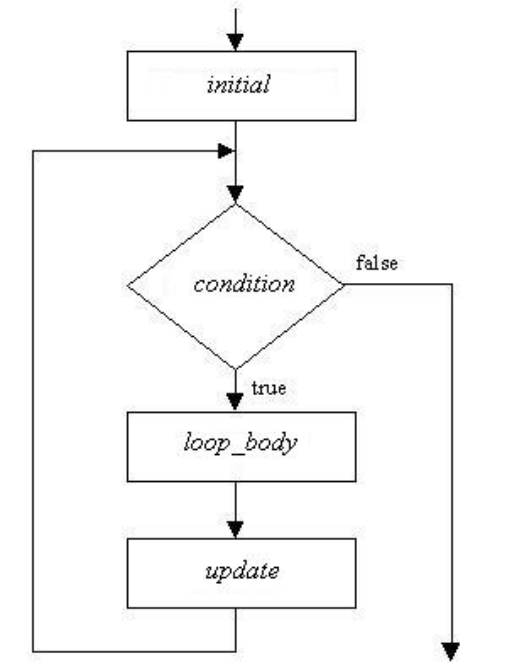
حلقه for احتمالا متداولترین حلقه در کلیه زبان های برنامه نویسی است. وقتی بخشی از کد به تعداد معینی تکرار شود بکار برده می شود. یک عدد صحیح به عنوان شمارنده حلقه برای شمارش تعداد دفعات اجرای حلقه استفاده می شود. در دستور باید تعیین شود شمارنده از کجا شروع کند، کی متوقف شود و چقدر افزیش یا کاهش پیدا کند.

در C++ فرم کلی حلقه for به صورت زیر است:

```
For (initial; condition; update)
   {
   loop_body
   }
```

قسمت های initial، condition و update همگی عبارت هستند و با هم درون یک جفت پرانتز قرار می گیرند و توسط علامت سمیکولن از هم جدا می شوند. loop_body یا بدنه حلقه دستورات داخل حلقه است که باید تکرار شود و می تواند یک دستور ساده یا دستور ترکیبی باشد. اگر بدنه حلقه فقط شامل یک دستور باشد نیازی به آکولاد نیست. اما اگر بیشتر از یکی باشد باید درون آکولاد محصور شود تا به عنوان بلاکی از کد دیده شود.

حلقه for طبق دیاگرام زیر اجرا می شود:



همانطور که دیده می شود ابتدا مرحله initial اجرا می شود. سپس condition بررسی می شود. اگر شرط برقرار نباشد حلقه به پایان می رسد و اجرا از دستور بعدی حلقه ادامه پیدا می کند. اگر شرط درست باشد loop_body اجرا می شود. سپس قسمت update اجرا می شود.

قسمت initial تنها یکبار در ابتدای ورود به حلقه اجرا می شود و معمولا برای اعلان و مقداردهی اولیه شمارنده ها و متغیرهای داخل حلقه استفاده می شود. قسمت condition عبارت شرطی است و هربار قبل از تکرار بدنه حلقه ارزیابی می شود. اگر در آغاز غلط باشد بدنه حلقه اصلا اجرا نخواهد شد. قسمت update در هربار تکرار حلقه اجرا می شود و معمولا برای افزایش یا کاهش شمارنده حلقه استفاده می شود.

نکته. قسمت های initial، condition و update هر کدام می توانند حذف شوند.
نکته. اگر قسمت شرط حذف شود همیشه درست فرض می شود.
نکته. برای خوانائی آکولادها زیر هم قرار می گیرند و دستورات داخل آنها سه space جلوتر نوشته می شود.

مثال. قطعه کد زیر اعداد ۱ تا ۲۰ را نمایش می دهد.

```
for (count = 1; count <= 20; count++)
   cout<< count <<endl;
```

با وجودیکه کاربرد اصلی دستور for برای حلقه های با تکرار معلوم است کارهای دیگری را هم می توان با آن انجام داد.

مثال. در حلقه زیر چون قسمت شرط حذف شده است همیشه درست درنظر گرفته می شود و عبارت Hello را دائما چاپ می کند.

```
for ( ; ; )
   cout << "Hello" << endl;
```

مثال. برنامه زیر کاراکترهای اسکی را نمایش می دهد.

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

int main()
{
  for(int i = 0; i < 128; i++)
    if (i != 26)  // ANSI Terminal Clear screen
      cout << " value: " << i
           << " character: "
           << char(i) // Type conversion
           << endl;
  return 0;
}
```

توجه کنید که متغیر i درون for اعلان شده است و تنها در بدنه حلقه قابل استفاده است.

معمولا شمارنده حلقه به صورت صعودی افزایش پیدا می کند ولی می توان حلقه هائی داشت که متغییر شمارنده آن به صورت های دیگری تغییر می کند.

مثال. در حلقه زیر شمارنده حلقه هر بار یک واحد کم می شود.

```
for (count = 100; count > 0; count--)
```

مثال. در حلقه زیر شمارنده حلقه هر بار 5 واحد زیاد می شود.

```
for (count = 0; count < 1000; count += 5)
```

مثال. قسمت initial نه تنها برای مقداردهی اولیه بلکه برای اعمال دیگر هم می تواند بکار برود.

```
count = 1;
for (cout << count ; count < 1000; count++);
```

مثال. در قسمت شرط حلقه می توان از عملگرهای رابطه ای و منطقی استفاده کرد.

```
for (count = 0; count < 1000 && array[count] != 0; )
    cout << array[count++];
```

مثال. بدنه حلقه می تواند یک جمله پوچ باشد. دستور زیر عناصر یک آرایه را برابر با عدد ۵۰ می کند.

```
for (count = 0; count < 1000; array[count++] = 50)
    ;
```

مثال. عملگر کاما را می توان برای مقداردهی یا بروزرسانی دو متغیر در دستور for استفاده کرد.

```
for (i = 0, j = 999; i < 1000; i++, j--)
    b[j] = a[i];
```

مثال. تا وقتی که کاربر عدد ۹۹ را وارد کند حلقه تکرار می شود.

```
int nbr = 0;
for ( ; nbr != 99; )
   cin >> nbr;
```

for تودرتو

حلقه ها مانند دستورات شرطی می توانند تودرتو باشند. یک دستور for می تواند درون دستور دیگری قرار بگیرد. در این حالت حلقه داخلی به تعداد تکرار شمارنده خود ضرب در شمارنده حلقه بیرونی تکرار می شود.

مثال. دستور cout درون حلقه داخلی 5×3=15 بار تکرار می شود.

```
for (int i = 1 ; i < 3 ; i++ )
   {
   for(int j=0 ; j < 4 ; j++ )
      {
      cout << "This is inner loop " <<  j ;
      cout << " of outer loop " << i << endl;
      }
    }
```

مثال. برنامه زیر با کاراکتر ستاره (*) یک مستطیل 20×3 رسم می کند.

```
#include <iostream.h>
int main()
{
    int col=3, row=20;
    for ( ; row > 0; row--)
      {
       for ( ; col > 0; col--)
          cout << "*";
       cout << endl;
      }
    return 0;
}
```

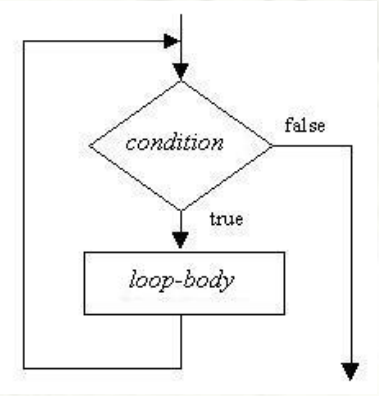
حلقه while

حلقه while بلاکی از کد را تا زمانی که شرط معینی true است (غیر صفر) اجرا می کند. شکل کلی آن به صورت زیر است:

```
while (condition)
   {
   loop_body
   }
```

condition یک عبارت شرطی است. اگر این عبارت false (یا صفر) باشد حلقه به پایان می رسد و دستور بعد از while اجرا می شود. body_loop دستوراتی است که تا زمان درست بودن شرط تکرار می شوند. اگر بدنه حلقه تنها یک دستور باشد نیازی به آکولاد نیست.

دیاگرام زیر نحوه اجرای حلقه while را نشان می دهد. توجه کنید که شرط در ابتدای حلقه بررسی می شود و اگر false باشد وارد حلقه نمی شود. اگر شرط true باشد بدنه حلقه اجرا می شود.



مثال. قطعه کد زیر اعداد ۱ تا ۲۰ را نمایش می دهد.

```
int count = 1;
while (count <= 20) {
   cout<< count <<endl;
   count++;
   }
```

حلقه while مانند یک حلقه for بدون قسمت های initial و update است: بنابراین هرکاری که با حلقه for می شود انجام داد با حلقه while هم می شود. اگر مقداردهی اولیه و افزایش متغیرها درون حلقه موردنیاز است حلقه for را بهتر است استفاده کنید. اگر حلقه while را استفاده می کنید مقداردهی اولیه موردنیاز قبل و افزایش درون حلقه باید انجام بگیرد.

while (*condition*) معادل است با for (; *condition* ;)

مثال. حلقه زیر تا زمان وارد شدن عدد ۹۹ ورودی دریافت می کند. عدد ۹۹ که انتهای داده ورودی را معین می کند sentinel نامیده می شود.

```
int nbr=0;
while (nbr <= 99)
    cin >> nbr ;
```

While تودرتو

مشابه دستورات if و for دستور while هم می تواند تودرتو باشد.

مثال. برنامه زیر ۵ عدد که باید مابین ۱ تا ۱۰ باشد از ورودی دریافت می کند.

```
#include <iostream.h>
main() {
   int array[5];
   int ctr = 0,
   nbr = 0;
   cout << "This program prompts you to enter 5 numbers" <<endl;
   cout << "Each number should be from 1 to 10" << endl;
   while ( ctr < 5 ) {
     nbr = 0;
     while (nbr < 1 || nbr > 10) {
        cout << endl << "Enter number " << ctr + 1 << "of 5: ";
        cin >> nbr;
        }
      array[ctr] = nbr;
      ctr++;
      }
    for (ctr = 0; ctr < 5; ctr++)
       cout << "Value " << ctr + 1 << " is " << array[ctr] << endl;
    return 0;
}
```

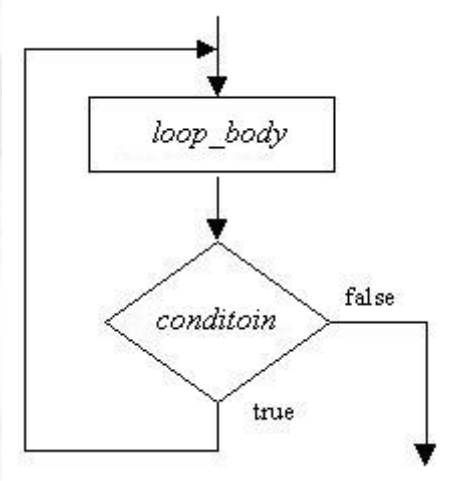
حلقه های do-while

سومین ساختار حلقه در C++ حلقه do- while است که بلاکی از کد را تا زمانی که شرط معینی true است اجرا می کند. حلقه do-while تقریبا مشابه حلقه while است با این تفاوت که شرط حلقه در انتهای حلقه است برخلاف حلقه while که در ابتدای قرار دارد. ساختار کلی آن به شکل زیر است:

```
do {
    loop_body
} while (condition);
```

condition یک عبارت شرطی است و تا وقتی که درست (یا غیرصفر) باشد بدنه حلقه تکرار می شود. چون شرط در انتها بررسی می شود بدنه حلقه حداقل یکبار اجرا می شود. اگر شرط برقرار نباشد حلقه به پایان می رسد و کنترل به دستور بعد از do-while منتقل می شود. بدنه حلقه می تواند یک دستور ساده باشد که در اینصورت نیازی به آکولاد نمی باشد.

دیاگرام حلقه do-while به صورت زیر است:



مثال. قطعه کد زیر اعداد ۱ تا ۲۰ را نمایش می دهد.

```
int count = 1;
do{
  cout<< count <<endl;
  count++;
} while (count <= 20);
```

مثال. حلقه زیر تا زمان وارد شدن عدد ۹۹ ورودی دریافت می کند.

```
Do {
    cin >> nbr ;
} while (nbr <= 99);
```

حلقه های do-while هم مشابه حلقه های for و while می توانند تودرتو باشند. دقت کنید که حلقه باید کاملا درون حلقه دیگر باشد و نباید overlap باشد.

دستورات break و continue

در C++ دو دستور وجود دارد که باعث ایجاد وقفه در اجرای عادی حلقه می شود؛ break و continue. دستور break را قبلا در switch آشنا شدید. به طور مشابه این دستور در حلقه های for، while و do-while باعث می شود کنترل بلافاصله از بدنه حلقه خارج شده به دستور بعد از حلقه منتقل شود.

مثال. در برنامه زیر به کاربر اجازه داده می شود ۱۰۰ عدد را وارد کند یا برای پایان یک عدد منفی وارد کند.

```
#include <iostream.h>

int main() {
    cout << "Enter 100 positive numbers, or a "
         <<" negative number to abort.\n";

    int i;

    for (i = 1; i <= 100; ++i) {
        cout << "Enter the number #" << i << ": ";
        int n;
        cin >> n;
        if (n < 0)
            break;
    }

    if (i == 100)
        cout << "You are a real man.\n";
    else
        cout << "You stopped after " << i
             << " numbers, coward!\n";
}
```

دستور continue باعث پرش از روی دستورات بعد از خود و انتقال کنترل به ابتدای حلقه می شود.

مثال. برنامه زیر مضارب غیر 7 مابین اعداد 0 تا 100 نمایش می دهد.

```
#include <iostream.h>

int main() {
    for (int i = 0; i < 100; ++i) {
        // Skip the multiples of 7
        if ((i % 7) == 0)
            continue;

        cout << i << " ";
    }
}
```