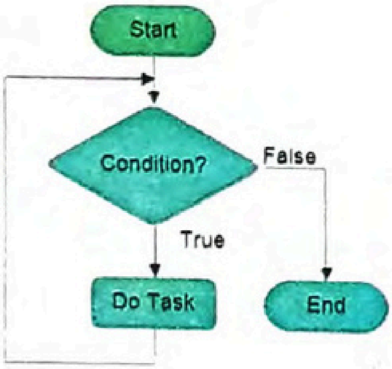
باب نمبر

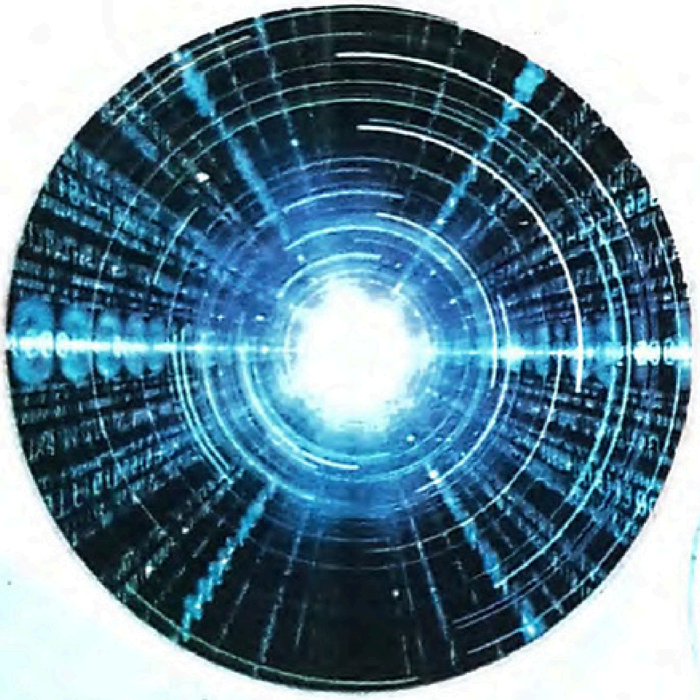
5

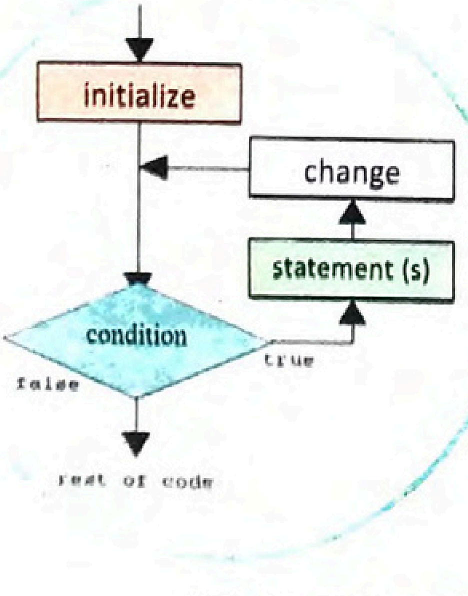
لوپ سٹرکچر

اس باب کی تکمیل کے بعد طلبہ اس قابل ہو جائیں گے کہ وہ:

- لوپ (Loop) کی ساخت تفصیل سے بیان کرسکیں۔
- *for* لوپ کی ساخت جان کر اس کا استعمال کرسکیں۔
- *while* لوپ کی ساخت جان کر اس کا استعمال کرسکیں۔
- *do-while* لوپ کی ساخت جان کر اس کا استعمال کرسکیں۔
- break اور continue سٹیٹمنٹس کے استعمال کو تفصیل سے بیان کرسکیں۔
- لوپ کی تمام اقسام میں فرق بیان کرسکیں۔
- نیسٹڈ (Nested) لوپ کے تصور کی وضاحت کرسکیں۔
- باب نمبر 1 میں زیر بحث فلو چارٹس کے لئے کوڈ لکھ سکیں۔


Start
Condition?
False
True
Do Task
End




initialize
change
statement (s)
condition
true
false
rest of code

5.1 تعارف (Introduction)

لوپ کمپیوٹر پروگرامنگ کا ایک سب سے طاقتور تصور ہے جو پروگرامرز کو اپنے پروگرام تخلیق کرنے میں مدد فراہم کرتا ہے جو مؤثر ہوں اور کاروباری ضروریات کو پورا کریں۔ لوپ ہدایات کا ایک ایسا تسلسل ہے جو کہ کسی خاص حالت (Condition) تک پہنچنے تک مستقل طور پر دہرایا جاتا ہے۔ مثال کے طور پر، اگر کوئی پروگرامر پے رول پروسیسنگ کے لئے ایسا پروگرام بناتا ہے جس میں کسی کمپنی میں تنخواہوں اور 1000 ملازمین کے الاؤنس کے حساب کی ضرورت ہوتی ہے۔ کمپیوٹر ایک لوپ کا استعمال کرتے ہوئے، تمام ملازمین کی تنخواہوں اور الاؤنس کی گنتی کرنا شروع کر دیتا ہے، جب تک کہ یہ عمل تمام ملازمین کے لئے پورا نہ کر دے۔ اسی طرح، کسی کمپنی میں موجود تمام ملازمین کی تنخواہوں کا پرنٹ آؤٹ حاصل کرنے کے لیے ایک لوپ کی ضرورت ہوتی ہے، تاکہ اس کے ذریعے تمام ملازمین کے لئے تنخواہوں کی سلپ تیار کی جاسکے اور اسے پرنٹ کیا جاسکے۔ اگر ایسی سرگرمی خود کی جائے تو، اس میں کئی گھنٹے اور دن لگیں گے۔ دوسری طرف لوپ کی مدد سے، وقت اور تھکا دینے والے کاموں کو تیزی سے انجام دیا جاتا ہے۔ کام کو درست انداز میں انجام دینے میں بھی وہ بہت مددگار ثابت ہوتے ہیں۔ C لینگویج میں مختلف قسم کے لوپ اسٹرکچر استعمال ہوتے ہیں۔

5.1.1 لوپ اسٹرکچر (Loop Structure)

ترتیب وار کنٹرول اسٹرکچر میں تمام اسٹیٹمنٹس کو ایک کے بعد دوسری، کی ترتیب میں چلایا جاتا ہے۔ جبکہ دوسری طرف، کنڈیشنل اسٹرکچر میں کسی بھی شرط کی بنیاد پر کسی اسٹیٹمنٹ کو چلایا جائے گا اور کسی کو نظر انداز کر دیا جائے گا۔ لیکن بعض حالتوں میں، ایک اسٹیٹمنٹ کو یا اسٹیٹمنٹس کے ایک بلاک کو ایک سے زیادہ مرتبہ چلانے کی ضرورت پڑتی ہے۔ اس مقصد کے لئے لوپ اسٹرکچر کا استعمال کیا جاتا ہے۔ ایسا کنٹرول اسٹرکچر جو ایک اسٹیٹمنٹ یا ایک سے زیادہ اسٹیٹمنٹس کو ایک سے زیادہ مرتبہ چلاتا ہے لوپنگ اسٹرکچر (Looping Structure) کہلاتا ہے۔ اس کو تکراری (Repetitive) اسٹرکچر بھی کہا جاتا ہے۔ C میں مختلف قسم کے لوپ دستیاب ہیں۔ لوپس بنیادی طور پر دو مقاصد کے لئے استعمال ہوتے ہیں۔

- کئی اسٹیٹمنٹس کو مقررہ تعداد تک بار بار چلانے کے لئے۔ مثال کے طور پر، ایک یوزر کمپیوٹر سکرین پر اپنا نام 20 بار پرنٹ کرنا چاہتا ہے۔
- کسی شرط کے True یا False نتیجہ کی بنیاد پر کسی اسٹیٹمنٹ یا ایک سے زیادہ اسٹیٹمنٹس کو چلانے کے لیے۔ مثال کے طور پر، ایک یوزر قدرتی اعداد کا ایک مجموعہ اس وقت تک پرنٹ کر سکتا ہے جب تک کہ ایک ویری ایبل X کی قیمت 50 تک نہیں پہنچ جاتی۔

ہمارے پاس C میں تین طرح کے لوپ ہیں۔ ان کا کام تقریباً ایک جیسا ہی ہے، تاہم، یہ مختلف حالات میں استعمال ہو رہے ہیں۔ ضرورت کے مطابق آپ کسی لوپ کا انتخاب کر سکتے ہیں:

- ***for*** لوپ
- ***while*** لوپ
- ***do-while*** لوپ

5.1.2 *for* لوپ

for لوپ کی مدد سے ایک یا ایک سے زیادہ اسٹیٹمنٹس کو جتنی دفعہ (Times) چاہیں چلا سکتے ہیں۔ اس لوپ کو پروگرامر بہت زیادہ استعمال کرتے ہیں۔ اس لوپ کا سانٹکس اور استعمال بہت آسان ہے۔

اس کا سانٹکس درج ذیل ہے:

```
for (initialization;condition;increment/decrement)
{
        Body of the loop
}
```

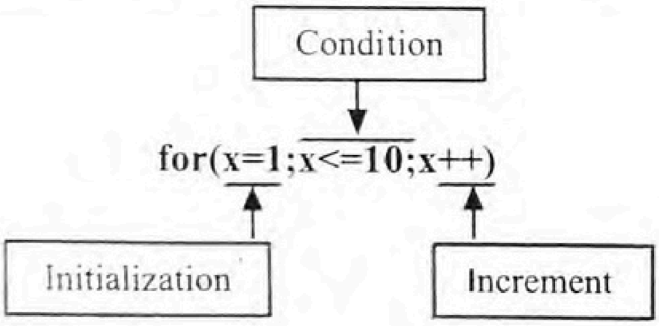
Initialization: اس سے مراد وہ عبارت (Expression) ہے جس سے کاؤنٹر ویری ایبل کی ابتدائی قیمت رکھی جاتی ہے۔ اس کا مطلب ہے کہ یہ حصہ لوپ میں صرف ایک بار چلتا ہے۔

condition: یہ کوئی بھی ریلیشنل ایکسپریشن ہو سکتا ہے۔ لوپ کے اندر تمام اسٹیٹمنٹس اس وقت چلائی جائیں گی جب تک اس condition کا نتیجہ True رہے، جیسے ہی اس condition کا نتیجہ False ہوا تو اسٹیٹمنٹ پھر نہیں چلائی جائیں گی اور کنٹرول لوپ کے بعد پروگرام میں اگلی اسٹیٹمنٹ پر منتقل ہو جائے گا جیسا کہ شکل نمبر 5.1 میں دکھایا گیا ہے۔

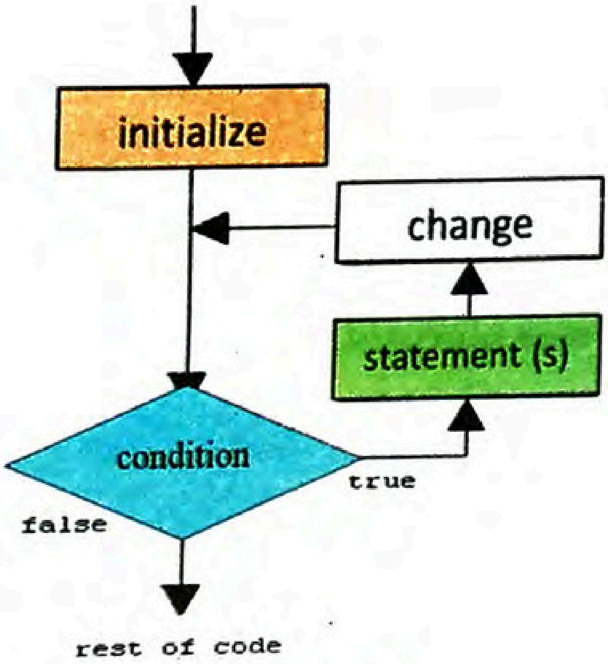
Increment/Decrement: لوپ کا یہ حصہ لوپ کے ہر تکرار (Iteration) کے بعد کاؤنٹر ویری ایبل کی قیمت میں تبدیلی کرتا ہے۔

Body of loop: یہ ایک اسٹیٹمنٹ یا اسٹیٹمنٹس کا ایک گروپ ہے جسے بار بار چلایا جاتا ہے یہ قوسین { } کے اندر بند ہوتا ہے۔

مثال کے طور پر:



تکرار کی تعداد لوپ کے اجزاء Initialization، Condition اور Increment/Decrement پر منحصر ہے۔ Initialization والا حصہ ایک دفعہ ہی چلتا ہے جب کنٹرول لوپ میں داخل ہوتا ہے۔ Initialization کے بعد Condition کو چیک کیا جاتا ہے۔ اگر اس کا نتیجہ True ہو تو، کنٹرول لوپ کی باڈی میں داخل ہو کر تمام اسٹیٹمنٹس پر عمل درآمد کرتا ہے۔ پھر Increment/Decrement حصہ پر عمل درآمد کیا جاتا ہے جو کاؤنٹر ویری ایبل کی قیمت کو تبدیل کرتا ہے۔ کنٹرول ایک بار پھر Condition والے حصے پر چلا جائے گا۔ یہ عمل بدستور اس وقت تک جاری رہے گا جب تک condition کا نتیجہ True رہتا ہے جیسے ہی یہ False ہو گا لوپ کو منقطع (Terminate) کر دیا جائے گا۔


initialize
change
statement (s)
condition
true
false
rest of code

شکل نمبر 5.1 for لوپ کی ترکیب

درج ذیل پروگرام for لوپ کے عمل کو ظاہر کرتا ہے۔

پروگرام 5.1 یہ پروگرام for لوپ کی مدد سے 1 سے لے کر 10 تک تمام اعداد کو پرنٹ کرے گا۔

```
# include <stdio.h>
void main()
  {
        int x;
        for (x=1;x<=10;x++)
        printf("%d\t",x);
}
```

آؤٹ پٹ

```
1    2    3    4    5    6    7    8    9    10
```

پروگرام 5.2 یہ پروگرام 10 سے لیکر 1 تک تمام اعداد کو (الٹی ترتیب میں) پرنٹ کرے گا۔

```
# include <stdio.h>
void main()
{
      int num;
      for (num=10;num>=1;--num)
      printf("%d\t",num);
}
```

آؤٹ پٹ

```
10    9    8    7    6    5    4    3    2    1
```

پروگرام 5.3 یہ پروگرام کوئی بھی عدد ان پٹ کروائے گا اور پھر اس عدد کا 10 تک پہاڑہ اسکرین پر پرنٹ کرے گا۔

```
# include <stdio.h>
void main()
{
        int tab,n,x;
        printf("\nEnter number for table: ");
        scanf("%d",&n);
        for(x=1;x<=10;x++)
        printf("\n%d X %d = %d",n,x,n*x);
}
```

آؤٹ پٹ

```
Enter number for table: 5
        5 × 1 = 5
        5 × 2 = 10
        5 × 3 = 15
        5 × 4 = 20
        5 × 5 = 25
        5 × 6 = 30
        5 × 7 = 35
        5 × 8 = 40
        5 × 9 = 45
        5 × 10 = 50
```

پروگرام 5.4 یہ پروگرام 1 سے لیکر 20 تک تمام طاق اعداد کا حاصل جمع معلوم کر کے اسکرین پر پرنٹ کرے گا۔

```
# include <stdio.h>
void main()
{
        int n, sum;
        sum = 0;
        for (n=1;n<=20;n+=2)
        {
        sum = sum+n;
        }
        printf("\nSUM = %d",sum);
}
```

آؤٹ پٹ

```
SUM = 100
```

5.1.3 while لوپ

while لوپ میں کنڈیشن کو پہلے جانچا جاتا ہے اور اگر اس کا نتیجہ True ہو تو while لوپ میں موجود اسٹیٹمنٹ چلا دی جاتی ہیں۔ یہ عمل اس وقت تک جاری رہتا ہے جب تک کنڈیشن کا نتیجہ False نہ ہو جائے۔ جب کنڈیشن کا نتیجہ False ہو جائے تو کنٹرول لوپ کے باہر آنے والی پہلی اسٹیٹمنٹ پر منتقل ہو جاتا ہے۔ اس لوپ کا استعمال ان صورتوں میں زیادہ موزوں رہتا ہے جب پہلے سے معلوم نہ ہو کہ لوپ نے مطلوبہ نتیجے کے لیے کتنی مرتبہ چلنا ہے۔ while لوپ کا سائنٹکس درج ذیل ہے۔

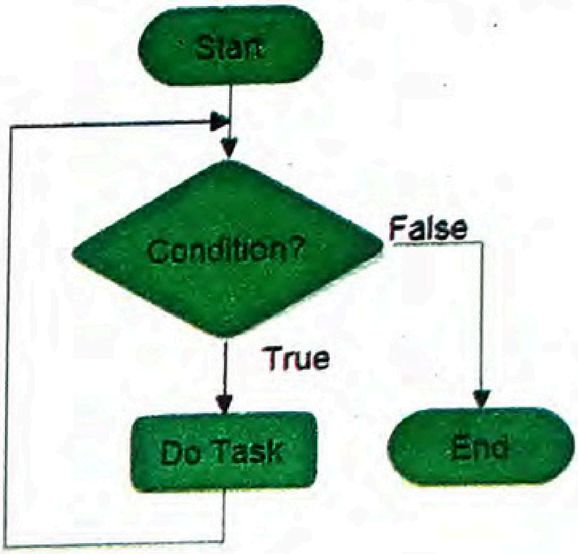
سائنٹکس:

```
while(condition)
{
Body of the loop
}
```

Condition: یہ ایک ریلیشنل ایکسپریشن ہوتا ہے۔ لوپ کی باڈی صرف اس صورت میں چلے گی جب condition کا نتیجہ True ہو گا اور اگر نتیجہ False ہوا تو پھر لوپ کی باڈی قطعاً نہیں چلے گی۔

Body of Loop: یہ قوسین { } کے درمیان اسٹیٹمنٹ یا اسٹیٹمنٹس کا ایک گروپ ہے جسے بار بار چلانا ہے۔

نوٹ: لوپ کا استعمال کرتے وقت اس بات کا خیال رکھنا ضروری ہے کہ لوپ ویری ایبل کو لوپ کے اندر ہر تکرار (Iteration) پر انکریمنٹ یا ڈیکریمنٹ کر دیا جائے۔ تاکہ لوپ کی کنڈیشن کسی بھی وقت False ہو کر لوپ کو ختم کر سکے ورنہ لوپ غیر معینہ مدت تک چلتی رہے گی۔ لوپ کا کام کرنا شکل نمبر 5.2 میں دکھایا گیا ہے۔



شکل نمبر 5.2 while لوپ کا کام کرنے کا انداز

درج ذیل پروگرام while لوپ کے عمل کو ظاہر کرتا ہے۔

پروگرام 5.4 یہ پروگرام while لوپ کی مدد سے "Pakistan" پانچ دفعہ سکرین پر پرنٹ کرے گا۔

```
# include <stdio.h>
void main()
{
        int n;
        n=1;
        while(n<=5)
        {
        printf("Pakistan\n");
        n++;
        }
}
```

آؤٹ پُٹ

```
Pakistan
Pakistan
Pakistan
Pakistan
Pakistan
```

پروگرام 5.6 یہ پروگرام while لوپ کی مدد سے 1 سے لے کر 19 تک طاق اعداد کو پرنٹ کرے گا۔

```
# include <stdio.h>
void main()
{
      int n=1; s=0;
      while(s<=100)
      {
      s=s+n;
      printf("%d\n",n);
      n+=2;
      }
}
```

آؤٹ پٹ

```
1
3
5
7
9
11
13
15
17
19
```

پروگرام 5.7 یہ پروگرام while لوپ کی مدد سے ذیل میں دیئے گئے عددی سلسلے (Series) کا نتیجہ نکال کر اسے سکرین پر پرنٹ کرے گا۔

2+4+6+...+20

```
# include <stdio.h>
void main()
{
      int c, sum=0;
      c=2;
      while(c<=20)
      {
      sum=sum+c;
      c=c+2;
      }
      printf("Sum=  %d",sum);
}
```

آؤٹ پٹ

```
SUM = 110
```

5.1.4 do-while لوپ

do-while لوپ اس وقت تک اسٹیٹمنٹ کو چلاتی رہتی ہے جب تک شرط کا نتیجہ True رہتا ہے۔ اس لوپ میں، Condition لوپ کی باڈی کے بعد لکھی جاتی ہے۔ یہ ایسی صورت حال میں استعمال ہوتی ہے جب لوپ کی باڈی کو بہر صورت کم از کم ایک بار چلانا مقصود ہو چاہے شرط کا نتیجہ پہلی بار غلط (False) ہو جائے۔ do-while لوپ کو شکل نمبر 5.3 میں دکھایا گیا ہے۔

نوٹ: do-while لوپ کا استعمال کرتے وقت اس بات کا خیال رکھنا چاہیئے کہ پہلے، لوپ کے اندر موجود اسٹیٹمنٹ چلائی جاتی ہیں اور پھر شرط کا جائزہ لیا جاتا ہے، اگر نتیجہ درست ہو جائے تو کنٹرول دوبارہ "do" پر چلا جاتا ہے تاکہ اسے بار بار چلایا جا سکے۔ ایسا بار بار ہوتا ہے جب تک کہ شرط کا نتیجہ غلط (False) نہ ہو جائے۔ ایک بار شرط false ہو جائے تب کنٹرول پروگرام میں اگلی اسٹیٹمنٹ پر منتقل ہو جاتا ہے۔

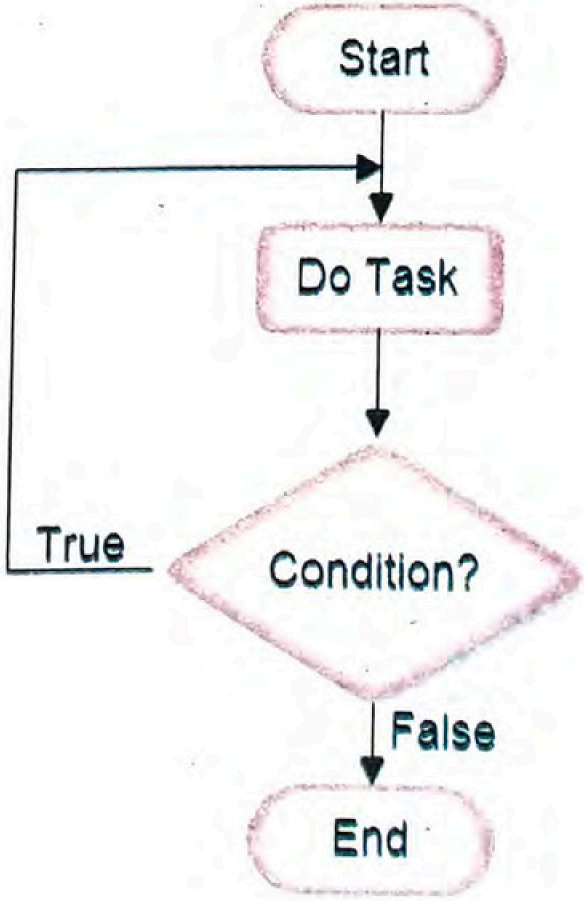
سائنٹکس :

```
do
{
Body of the loop
}
while(condition);
```

do: یہ وہ کلیدی لفظ ہے جو لوپ کے آغاز کی طرف اشارہ کرتا ہے۔

Condition: یہ ریلیشنل ایکسپریشن ہے۔ اگر شرط کا نتیجہ درست ہو تو باڈی کے اندر اسٹیٹمنٹس پر عمل درآمد کیا جاتا رہے گا۔ اگر شرط کا نتیجہ False ہو جائے، تو باڈی کے اندر اسٹیٹمنٹس صرف ایک بار چلیں گی کیونکہ اس میں شرط آخر میں چیک کی جاتی ہے۔ لوپ کو سیمی کالن کے ساتھ ختم ہونا ضروری ہے۔

Body of the loop: یہ قوسین { } کے درمیان اسٹیٹمنٹ یا اسٹیٹمنٹ کا ایک گروپ ہے جس کو بار بار چلانا ہے۔



پروگرام 5.8 یہ پروگرام do-while لوپ کی مدد سے گنتی کے پہلے دس اعداد کو پرنٹ کرے گا۔

```
# include <stdio.h>
void main()
{
     int n;
     n=1;
     do
     {
     printf("%d\n",n);
     n++;
     }
     while(n<=10);
}
```

آؤٹ پٹ

```
1
2
3
4
5
6
7
8
9
10
```

پروگرام 5.9 یہ پروگرام do-while لوپ کی مدد سے یہ معلوم کرے گا کہ اِن پُٹ کیا گیا عدد طاق ہے یا جفت۔

```
# include <conio.h>
# include <stdio.h>
void main()
{
    int num;
    char choice;
    do
    {
       printf("\nEnter a number to find even or odd");
        scanf("%d", &num);
        if((num%2)==0)
              printf("\nNumber is even");
       else
              printf("\nNumber is odd");
     printf("\nDo you want to check another");
     choice = getche();
     }
       while(choice == 'y');
       printf("\nThank You");
}
```

آؤٹ پٹ

```
Enter a number to find
even or odd
102
Number is even
```

5.1.5 break اسٹیٹمنٹ

بریک اسٹیٹمنٹ جب لوپ کی باڈی کے اندر استعمال ہو تو یہ لوپ کو منقطع کر دے گی اور لوپ مطلوبہ تکرار (Iteration) مکمل کئے بغیر ہی ختم ہو جائے گی۔

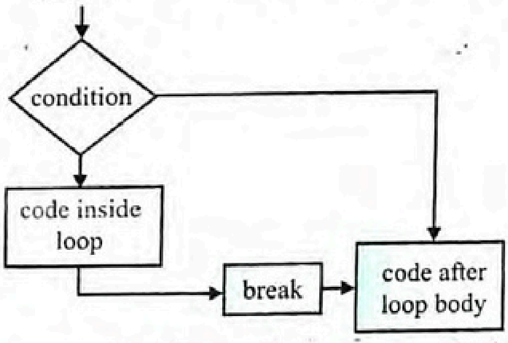
سائنٹکس:

```
break;
```

break اسٹیٹمنٹ کو ان دو صورتوں میں استعمال کیا جاتا ہے۔

فوری طور پر لوپ سے باہر آنے کے لئے break اسٹیٹمنٹ کا استعمال کریں۔ جب بھی کسی لوپ کے اندر break اسٹیٹمنٹ کا سامنا کرنا پڑتا ہے تو، کنٹرول اسے ختم کر کے براہ راست لوپ سے نکل آتا ہے۔ لوپ کے اندر اسے کنڈیشنل اسٹیٹمنٹ کے ساتھ استعمال کیا جاتا ہے، تاکہ یہ صرف کسی خاص condition میں ہی چل سکے۔

یہ switch-case سوئچ کیس کنٹرول اسٹرکچر میں case block کے اندر استعمال ہوتا ہے۔ عام طور پر switch-case میں تمام case کے بعد break اسٹیٹمنٹس لکھی جاتی ہے تاکہ اس کے بعد کے case پر عملدرآمد سے بچا جا سکے۔ جب بھی اس کا سامنا switch-case بلاک میں ہوتا ہے، کنٹرول switch-case باڈی سے باہر نکل آتا ہے۔ break اسٹیٹمنٹ کا کام شکل نمبر 5.4 میں دکھایا گیا ہے۔



شکل نمبر 5.4 break اسٹیٹمنٹ کا کام کرنے کا طریقہ کار

پروگرام 5.10 یہ پروگرام for لوپ کو استعمال کر کے یوزر سے کی بورڈ کی مدد سے ایک عدد ان پٹ کرے گا اگر نمبر 0 سے بڑا ہوا تو اسے پرنٹ کر کے اگلا عدد اِن پٹ کر دیا جائے گا۔ اگر یوزر نے 0 یا اس سے چھوٹا یعنی منفی عدد اِن پٹ کیا تو لوپ منقطع ہو جائے گا۔ یاد رہے کہ یہاں break اسٹیٹمنٹ استعمال ہوئی ہے۔

```
# include <stdio.h>
void main()
{
        int x, num;
        for(x=1;x<=5;x++)
        {
        printf("\nEnter a number: ");
        scanf("%d", &num);
        if(num<=0)
        break;
        printf("\nYou entered  %d\n ",num);
        }
        printf("\n OK");
}
```

آؤٹ پُٹ

```
Enter a number: 3
You entered 3
Enter a number: 0
OK
```

مذکورہ بالا پروگرام میں لوپ کے اندر break اسٹیٹمنٹ استعمال کی گئی ہے۔ کاؤنٹر ویری ایبل x بتاتا ہے کہ لوپ کو پانچ بار چلنا چاہئے۔ لیکن یہ صرف دو بار ہی چلے گی۔ پہلی بار میں، سکرین پر 3 پرنٹ ہو گا۔

دوسری بار میں printf( ) اسٹیٹمنٹ سوال پرنٹ کرے گی اور یوزر نے 0 اِن پٹ کیا ہے تو اس سے کنٹرول break اسٹیٹمنٹ پر چلا جائے گا۔ break اسٹیٹمنٹ چلنے سے کنٹرول لوپ سے باہر آنے والی پہلی اسٹیٹمنٹ پر منتقل ہو جائے گا۔ اس لیے پیغام (Message) صرف ایک بار ہی پرنٹ ہو گا۔

5.1.6 continue اسٹیٹمنٹ

continue اسٹیٹمنٹ کنٹرول کو لوپ کے آغاز پر واپس منتقل کر دیتی ہے۔ یہ لوپ کی باڈی کے اندر استعمال ہوتی ہے۔ جب بھی، ایک لوپ کے اندر continue اسٹیٹمنٹ کا سامنا کرنا پڑتا ہے تو، کنٹرول اگلے تکرار کے لئے لوپ کے آغاز پر براہ راست منتقل ہو جاتا ہے، موجودہ iteration کے لیے لوپ کی باڈی کے اندر اسٹیٹمنٹ کی تعمیل کو چھوڑ دیتا ہے۔ continue اسٹیٹمنٹ کو شکل نمبر 5.5 میں دکھایا گیا ہے۔

سائنٹکس:

```
continue;
```

continue اسٹیٹمنٹ break اسٹیٹمنٹ کے بالکل مخالف ہے۔ اسے بائی پاس اسٹیٹمنٹ بھی کہا جاتا ہے۔

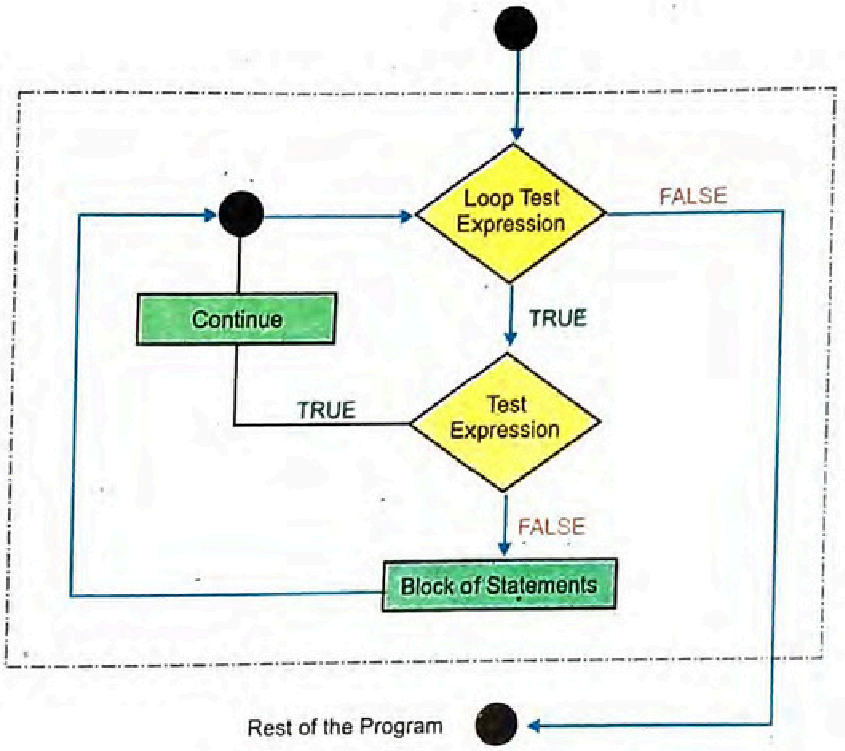
پروگرام 5.11 ایک پروگرام لکھیں اور اس میں while لوپ کے اندر continue اسٹیٹمنٹ کے استعمال کو سمجھیں۔

```
#include <stdio.h>
void main()
{
      int c;
      c=1;
      while(c<=5)
      {
      printf("Pakistan\n");
      c++;
       continue;
       printf("Islamabad");
       printf("Peshawar");
       }
}
```

آؤٹ پٹ

```
Pakistan
Pakistan
Pakistan
Pakistan
Pakistan
```

مذکورہ پروگرام میں، لوپ کی پہلی 3 اسٹیٹمنٹ 5 مرتبہ چلائی جائیں گی۔ جبکہ continue اسٹیٹمنٹ کے بعد کی دو اسٹیٹمنٹ نہیں چلیں گی۔ کیونکہ ہر دفعہ جب continue اسٹیٹمنٹ چلتی ہے تو یہ کنٹرول کو لوپ کی باڈی کے بالکل آغاز پر واپس بھیج دیتی ہے۔


Loop Test Expression
FALSE
TRUE
Continue
TRUE
Test Expression
FALSE
Block of Statements
Rest of the Program

شکل نمبر 5.5 continue اسٹیٹمنٹ کا طریقہ کار

5.1.7 مختلف لوپ سٹرکچرز میں فرق

ذیل میں for، while اور do-while کے فرق کو بیان کیا گیا ہے۔

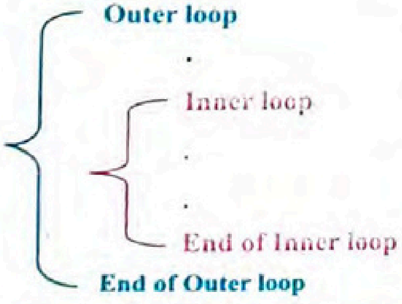
for لوپ: اس کے استعمال کو ترجیح دی جاتی ہے جب پروگرامر پہلے سے جانتا ہو کہ لوپ کی باڈی پر کتنی بار عملدرآمد کیا جائے گا۔

while لوپ: یہ ایسے حالات میں استعمال کرنے کے لئے موزوں ہے جہاں پروگرامر کو پتہ نہیں ہوتا ہے کہ لوپ کی باڈی کو کتنی مرتبہ چلنا ہے۔ اِس لوپ میں کنڈیشن لوپ کی باڈی سے پہلے آتی ہے۔ پہلے، کنڈیشن کی جانچ کی جاتی ہے اور پھر مطلوبہ نتیجہ حاصل کرنے کے لئے لوپ کی باڈی کے اندر اسٹیٹمنٹ کو چلایا جاتا ہے۔

do-while لوپ: یہ ایسے حالات میں استعمال کرنے کے لئے موزوں ہے جہاں پروگرامر چاہتا ہے کہ لوپ کو کم سے کم ایک بار ضرور چلایا جائے۔ do-while لوپ میں، لوپ کی باڈی ٹیسٹ کی condition سے پہلے آتی ہے۔ لوپ کی باڈی کو چلایا جاتا ہے اور پھر اس کی شرط کی جانچ کی جاتی ہے۔ اس لوپ میں، شرط سیمی کالن یعنی اسٹیٹمنٹ ٹرمینیٹر کے ساتھ ختم کر دی جاتی ہے۔

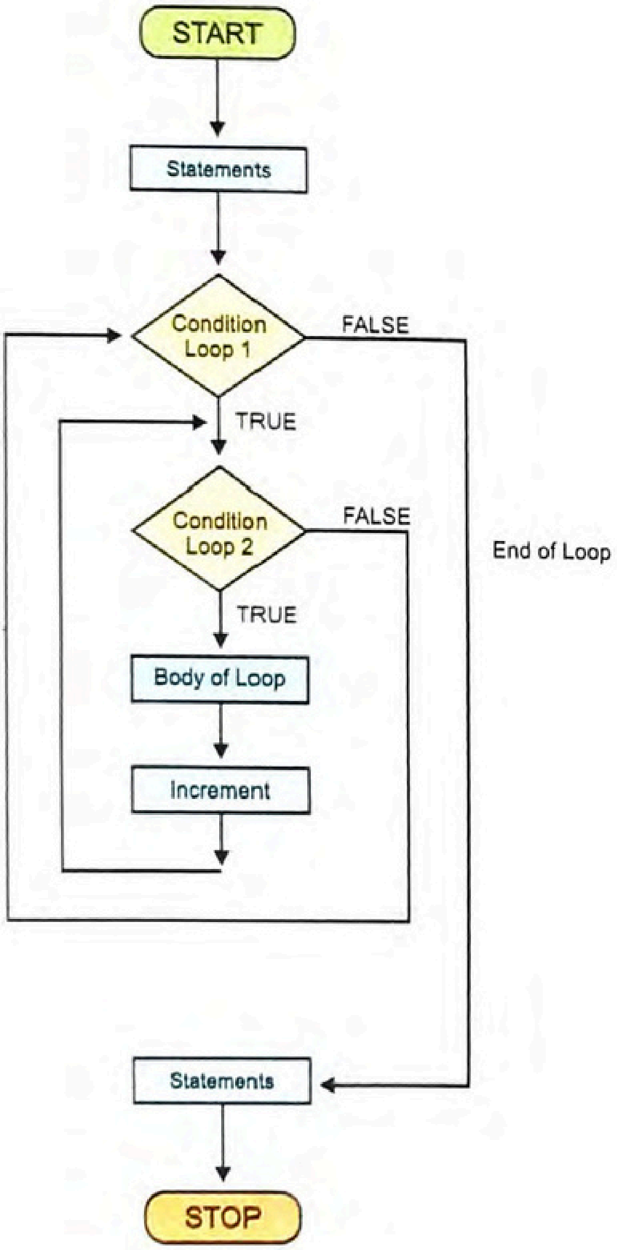
5.1.8 نیسٹڈ لوپ

ایک لوپ کے اندر دوسری لوپ کو نیسٹڈ لوپ کہتے ہیں۔ بیرونی لوپ کے اندر اندرونی لوپ مکمل طور پر بند ہوتی ہے۔ ذیل میں دی گئی شکل نیسٹڈ لوپ کے استعمال کی وضاحت کرتی ہے۔


Outer loop
Inner loop
End of Inner loop
End of Outer loop

نیسٹڈ لوپ کا سائنٹکس ذیل میں واضح کیا گیا ہے۔

```
for (initialize counter; test condition; ++ or --)
  {
        for (initialize counter; test condition; ++ or --)
        {
        .        // inner for loop
        }
        .        // outer for loop
  }
```


START
Statements
Condition Loop 1
FALSE
TRUE
Condition Loop 2
FALSE
TRUE
Body of Loop
Increment
End of Loop
Statements
STOP

شکل نمبر 5.6 نیسٹڈ لوپ

پروگرام 5.12 یہ پروگرام نیسٹڈ لوپ کی مدد سے آؤٹ پٹ دکھائے گا۔

```
# include <stdio.h>
void main()
{
        int i,j;
        for(i=1;i<=5;i++)
        {
        for(j=1;j<=i;j++)
        printf("%d\t",j);
        }
        printf("\n");
        }
}
```

آؤٹ پٹ

```
1
1   2
1   2   3
1   2   3   4
1   2   3   4   5
```

5.1.9 باب 1 کے فلو چارٹس کے لیے کوڈز

پروگرام 5.13 یہ پروگرام ایک عدد کا Factorial معلوم کرے گا۔

```
#include <iostream>
using namespace std;
int main() {
  int i, n, factorial = 1;
  cout<<"Enter a positive integer: ";
  cin>>n;
  for (i = 1; i <= n; ++i) {
     factorial *= i;   // factorial = factorial * i;
  }
  cout<< "Factorial of "<<n<<" = "<<factorial;
              return 0;
}
```

آؤٹ پٹ

```
Enter a positive integer: 5
Factorial of 5 = 120
```

پروگرام 5.14 پروگرام 1 سے 100 تک طاق اعداد کو پرنٹ کرے گا۔

```
#include<iostream>
using namespace std;
int main()
{
      int i=1;
      cout<<"All Odd numbers from 1 to 100 "<<endl<<endl;
      while(i<=100)
      {
       if(i%2==1)
```

```
    {
      cout<<i<<"\t";
     }
    i++;
    }
    getch();
    return 0;
}
```

آؤٹ پٹ

```
All odd numbers from 1 to 100
1 3 5 7 9 11 13 15 17 19 21 23 25 27 2931 33 35 37 39 41 43 45 47 49 51 53
55 57 5961 63 65 67 69 71 73 75 77 79 81 83 85 87 89 91 93 95 97 99
```

پروگرام 5.15 یہ پروگرام 1 سے 50 کے درمیان جفت اعداد کا حاصل جمع معلوم کرے گا۔

```
#include <iostream>
using namespace std;
int main() {
  int i, n, sum = 0;
    Take input from user.
  cout << "Print sum of even numbers till : ";
  cin >> n;
  for(i = 1; i <= n; i++) {
      Check for even or not.
     if((i % 2) == 0) {
        sum += i;
     }
  }
  cout << endl << "Sum of even numbers from 1 to 50 " << n << " is : " << sum;
  return 0;
}
```

آؤٹ پٹ

```
Print sum of even numbers till: 50
Sum of even numbers from 1 to 50 is: 1275
```

اہم نکات

- لوپ ہدایات کا ایک سلسلہ ہے جو کسی خاص حالت تک پہنچنے تک بار بار چلایا جاتا ہے۔
- for لوپ ایک مخصوص تعداد میں ایک یا زیادہ اسٹیٹمنٹ کو چلاتی ہے۔
- while لوپ ایک یا زیادہ اسٹیٹمنٹ پر اس وقت تک عمل درآمد کرتا رہتا ہے جب تک condition کا نتیجہ True رہتا ہے۔
- do-while لوپ اس وقت تک اسٹیٹمنٹ کو چلاتی رہتی ہے جب تک شرط کا نتیجہ True رہتا ہے۔
- break اسٹیٹمنٹ جب لوپ کی باڈی کے اندر استعمال ہو تو کنٹرول کو لوپ سے باہر بھیج دیتا ہے۔
- Continue اسٹیٹمنٹ کنٹرول کو لوپ کے آغاز پر واپس منتقل کرتا ہے۔
- ایک لوپ کے اندر دوسرے لوپ کو نیسٹڈ لوپ کہتے ہیں۔

مشق

سوال نمبر 1۔ موزوں جواب کا انتخاب کریں۔

i. C میں کتنی قسم کے لوپ سٹرکچر ہیں؟

(الف) 4 (ب) 3 (ج) 2 (د) 6

ii. کون سی لوپ ہمیشہ کم از کم ایک مرتبہ ضرور چلتی ہے؟

(الف) do-while (ب) for (ج) while (د) nested

iii. کس لوپ میں شرط ہمیشہ لوپ کی باڈی سے پہلے آتی ہے؟

(الف) while loop (ب) do-while loop

(ج) for loop (د) (الف) اور (ج) دونوں

iv. کس لوپ میں شرط ہمیشہ لوپ کی باڈی کے بعد آتی ہے۔

(الف) for (ب) while (ج) do-while (د) None

v. ذیل میں دی گئی کون سی اسٹیٹمنٹ لوپ کی باڈی سے باہر نکلنے کے لیے استعمال ہوتی ہے؟

(الف) break (ب) terminate (ج) exit (د) (الف) اور (ب) دونوں

vi. اگر ایک لوپ کو دوسری لوپ کے اندر بند کر دیا جائے تو اسے کیا کہا جائے گا؟

(الف) counter loop (ب) complex loop

(ج) outer loop (د) nested loop

vii. کونسی اسٹیٹمنٹ کنٹرول کو لوپ کی باڈی کے بالکل آغاز میں لے جاتی ہے؟

(الف) continue (ب) break (ج) while (د) for

viii. جب پہلے سے یہ پتہ ہو کہ لوپ نے کتنی دفعہ چلنا ہے تو ایسی صورت میں کونسی لوپ استعمال کرنا چاہیئے؟

(الف) do-while (ب) for (ج) while (د) nested

سوال نمبر۔2 لوپنگ سٹرکچر کی مختلف اقسام پر مثالوں کی مدد سے بحث کریں۔

سوال نمبر۔3 do-while اور while لوپ میں مثالوں کی مدد سے تفصیل سے فرق بیان کریں۔

سوال نمبر۔4 C میں ایک پروگرام لکھیں جو 1 اور 100 کے درمیان آنے والے تمام جفت اعداد کا حاصل ضرب معلوم کرے۔ یاد رہے کہ while لوپ کا استعمال کریں۔

سوال نمبر5۔ for لوپ استعمال کرتے ہوئے ایک پروگرام لکھیں جو A سے لیکر Z تک تمام حروف تہجی پرنٹ کرے۔

سوال نمبر6۔ نیسٹڈ لوپ کی مدد سے ایک پروگرام لکھیں جو ذیل میں دکھائی گئی آؤٹ پٹ کو سکرین پر پرنٹ کرے۔

```
            a
         a  a
      a  a  a
   a  a  a  a
a  a  a  a  a
```

سوال نمبر7۔ for لوپ کو استعمال کر کے C میں ایک پروگرام لکھیں جو 1 سے لیکر 50 تک تمام طاق اعداد کا حاصل جمع معلوم کرے۔ سکرین پر پرنٹ کر سکے۔