باب 4

ڈیٹا اینڈ ریپیٹیشن

تدریسی مقاصد: (Student Learning Outcomes)

- ارے (Array) کے سٹرکچر کو سمجھنا
- یک سمتی ارے کو ڈکلیئر کرنا اور اس کا استعمال
- ویری ایبل کو ارے کے انڈیکس کے طور پر استعمال کرنا
- ارے میں قیمتیں لکھنا اور انھیں پڑھنا
- لوپ سٹرکچر کے تصور کی وضاحت کرنا
- یہ معلوم ہونا کہ لوپ سٹرکچر ان سب سے بنا ہے
  - For
  - انیشلائزیشن ایکسپریشن
  - ٹیسٹ ایکسپریشن
  - لوپ کی باڈی
  - انکریمنٹ / ڈیکریمنٹ ایکسپریشن
- نیسٹڈ لوپ کے تصور کی وضاحت
- ارے میں ڈیٹا درج کرنے اور پڑھنے کے لیے لوپس کا استعمال کرنا

تعارف (Unit Introduction)

کمپیوٹر پروگرام لکھتے ہوئے ہمیں ایسے حالات کا بھی سامنا کرنا پڑسکتا ہے جب ہمیں ڈیٹا کی بڑی مقدار کو پروسیس کرنا ہو۔اب تک ہم نے جتنے بھی طریقے پڑھے ہیں وہ ایسے حالات میں مناسب نہیں معلوم ہوتے۔اس لیے زیادہ ڈیٹا کو محفوظ اور پروسیس کرنے کے لیے ہمیں بہتر میکانزم کی ضرورت ہے۔ایک اور عام مسئلہ جس کا ہمیں سامنا کرنا پڑتا ہے وہ یہ ہے کہ ہم ہدایات کے ایک سیٹ کو بار بار لکھے بغیر ایک سے زیادہ مرتبہ کیسے دُہرائیں۔اس باب میں ہم C-لینگوئج کے وہ طریقے دیکھیں گے جو ہمیں ڈیٹا اور ریپٹیشن سے نمٹنے میں مدد دیں۔

4.1 ڈیٹا سٹرکچرز (Data Structures)

گذشتہ ابواب میں ہم نے پڑھا کہ کس طرح ڈیٹا کو ویری ایبلز میں محفوظ کیا جاتا ہے۔اگر ہمیں زیادہ مقدار میں ڈیٹا محفوظ اور پروسیس کرنا ہو جیسے 100 طالب علموں کے نمبر تو شاید ہمیں 100 ویری ایبلز ڈکلیئر کرنے پڑیں جو کہ ایک بہتر حل نہیں ہے۔اس طرح کے کاموں کے لیے ہائی لیول کی پروگرامنگ لینگوئجز میں ڈیٹا کو محفوظ کرنے اور ترتیب دینے کے لیے ڈیٹا سٹرکچرز ہوتے ہیں۔ڈیٹا سٹرکچر کی تعریف یہ ہے:

''ڈیٹا سٹرکچر ایک کنٹینر ہوتا ہے جو ڈیٹا آئٹمز کے مجموعے کو ایک خاص ترتیب میں محفوظ کرنے کے لیے استعمال ہوتا ہے۔''

C-پروگرامنگ لینگوئج میں مختلف ڈیٹا سٹرکچرز موجود ہیں۔ہم اس باب میں اِن میں سے ایک ڈیٹا سٹرکچر ارے (Array) کے بارے میں پڑھیں گے۔یہ سب سے زیادہ استعمال ہونے والے ڈیٹا سٹرکچرز میں سے ایک ہے۔

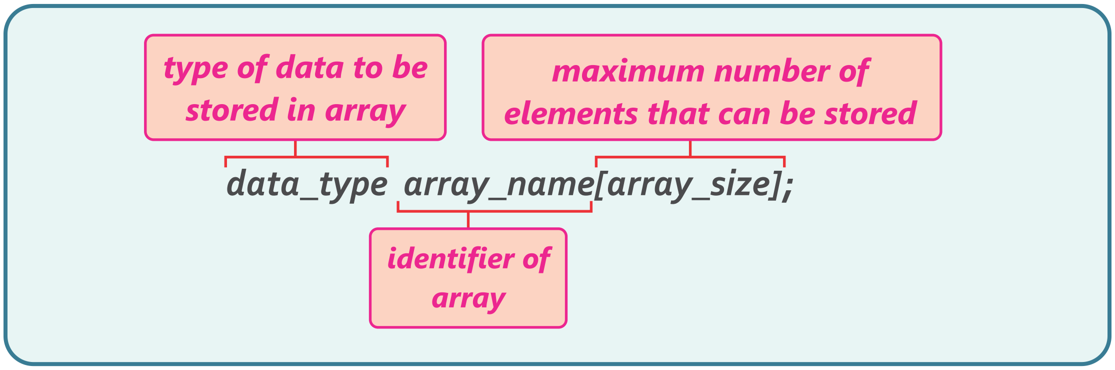
4.1.1 ارے (Array)

ارے ایک ڈیٹا سٹرکچر ہے جس میں ایک ہی ڈیٹا ٹائپ کی ایک سے زیادہ قیمتیں رکھی جا سکتی ہیں۔جیسے ایک int ارے میں بہت سی انٹیجر قیمتیں، ایک فلوٹ ارے میں بہت سی رئیل (Real) قیمتیں وغیرہ۔

ارے کی ایک اہم خصوصیت یہ ہے کہ یہ کمپیوٹر میموری میں تمام قیمتیں اکٹھی محفوظ کرتا ہے۔

4.1.2 ارے ڈکلیئریشن (Array Decleration)

C-لینگویج میں ارے کو اس طرح ڈکلیئر کیا جا سکتا ہے۔



اگر ہم چاہتے ہیں کہ ایک int ٹائپ کی ارے ہو جس میں ایک مزدور کی سات دن تک روزانہ کی اجرت رکھی جائے تو ہم اسے اس طرح ڈکلیئر کریں گے:

```
int daily_wage[7];
```

نیچے 20 طلبہ کے نمبر رکھنے کے لیے فلوٹ ٹائپ ارے کی ڈکلیئریشن کی مثال ہے۔

```
float marks[20];
```

4.1.3 ارے انیشلائزیشن (Array initialization)

پہلی مرتبہ ایک ارے میں قیمتیں رکھنا 'ارے انیشلائزیشن' کہلاتا ہے۔ ایک ارے کو ڈکلیئریشن کے وقت بھی انیشلائز کیا جا سکتا ہے اور بعد میں بھی۔ ڈکلیئریشن کے وقت ارے کو اس طریقے سے انیشلائز کر سکتے ہیں۔

data_type array_name[N] = {value1, value2, value3,........, valueN};

درج ذیل مثال میں سات بندوں کی قامت محفوظ کرنے کے لیے ایک فلوٹ ٹائپ ارے کو ڈکلیئر اور انیشلائز کیا گیا ہے۔

```
float height[7] = {5.7, 6.2, 5.9, 6.1, 5.0, 5.5, 6.2};
```

ذیل میں ایک اور مثال ہے جس میں انگریزی کے پانچ حروفِ علت (Vowels) محفوظ کرنے کے لیے ارے انیشلا ئز کی گئی ہے۔

```
char vowels[5] = {'a', 'e', 'i', 'o', 'u'};
```

اہم نوٹ:

اگر ہم ڈکلیئریشن کے وقت ارے کو انیشلا ئز نہیں کرتے تو ہمیں ایک ایک کر کے ارے کے ایلیمنٹس کو انیشلا ئز کرنا پڑتا ہے۔ اس کا مطلب یہ کہ پھر ہم ایک سٹیٹمنٹ میں ارے کے تمام ایلیمنٹس کو انیشلا ئز نہیں کر سکتے۔ یہ اس مثال سے واضح کیا گیا ہے۔

EXAMPLE CODE 4.1

```
void main()
{
      int array[5];
      array[5] = {10, 20, 30, 40, 50};
}
```

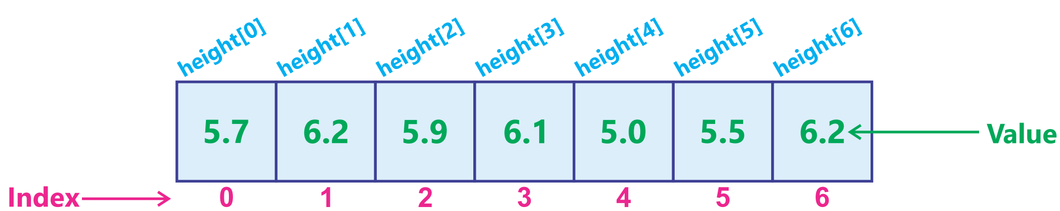
ERROR: initializing whole array after declaration not allowed

اوپر دی گئی مثال میں جب ہم ڈکلیئر کرنے کے بعد ارے کو ایک الگ سٹیٹمنٹ میں انیشلا ئز کرنے کی کوشش کرتے ہیں تو کمپائلر ایرر دیتا ہے۔

4.1.4 ارے ایلیمنٹس تک رسائی (Accessing Array Elements)

ارے کے ہر ایلیمنٹ کا ایک انڈیکس ہوتا ہے جس کو ارے کے نام کے ساتھ اس طرح ;array-name[index] لکھ کر ہم اس انڈیکس کے ڈیٹا تک رسائی حاصل کر سکتے ہیں۔

پہلے ایلیمنٹ کا انڈیکس 0 ہوتا ہے، دوسرے کا 1 اور آگے ایسے ہی چلتا ہے۔ لہٰذا height[0] ارے height کے پہلے ایلیمنٹ کا حوالہ دیتا ہے۔ height[1] دوسرے ایلیمنٹ کا اور اسی طرح آگے چلتا ہے۔ شکل 4.1 میں پچھلے سیکشن میں جو ارے height انیشلا ئز کی گئی ہے اس کی تصویر دی گئی ہے۔



شکل 4.1: ارے height کی گرافیکل (تصویر کے ذریعے) نمائندگی

4.1 پروگرامنگ ٹائم (Programming Time)

ایک پروگرام لکھیں جو پانچ لوگوں کی عمریں ایک ارے میں محفوظ کرے اور سکرین پر دکھائے۔

حل:

```
#include<stdio.h>
void main()
{
      int age[5];
      /* Following statements assign values at different
      indices of array age. We can see that the first value is
      stored at index 0 and the last value is stored at index 4
      */
      age[0] = 25;
      age[1] = 34;
      age[2] = 29;
      age[3] = 43;
      age[4] = 19;
      /* Following statement displays the ages of five persons
      stored in the array */
      printf("The ages of five persons are: %d, %d, %d, %d,
      %d", age[0], age[1], age[2], age[3], age[4]);
}
```

4.2 پروگرامنگ ٹائم (Programming Time)

ایک پروگرام لکھیں جو 4 مضامین کے حاصل کردہ نمبر صارف سے ان پٹ لے اور حاصل کردہ کُل نمبر سکرین پر دکھائے۔

حل:

```
#include<stdio.h>
void main()
{
      float marks[4], total_marks;
      printf("Please enter the marks obtained in 4 subjects:
      ");
      scanf("%f%f%f%f",&marks[0], &marks[1], &marks[2],
      &marks[3]);
      total_marks = marks[0] + marks[1] + marks[2] + marks[3];
      printf("Total marks obtained by student are %f",
      total_marks);
}
```

4.1.5 ویری ایبلز کا بطور ارے انڈیکس استعمال:

اریز کی ایک اہم خاصیت یہ ہے کہ ویری ایبلز کو بطور ارے انڈیکس استعمال کیا جا سکتا ہے جیسے درج ذیل پروگرام میں کیا گیا ہے۔

EXAMPLE CODE 4.2

```
#include<stdio.h>
void main()
{
	int array[5] = {10, 20, 30, 40, 50};
	int i;
	/* Following statements ask the user to input an index
	into variable i. */
	printf("Please enter the index whose value you want to
	display");
	scanf("%d", &i);
	/* Following statement displays the value of the array
	at the index entered by user. */
	printf("The value is %d", array[i]);
}
```

درج ذیل پروگرام ظاہر کرتا ہے کہ ہم ویری ایبل کی قیمت کو بدل سکتے ہیں۔اس کے بعد جہاں بھی ویری ایبل ہوگا وہاں اس کی نئی قیمت استعمال ہوگی۔

EXAMPLE CODE 4.3

```
#include<stdio.h>
void main()
{
	int array[5] = {10, 20, 30, 40, 50};
	int i = 2;
	/* Following statement displays value 30, as i contains
	2 and the value at array[2] is 30 */
	printf("%d", array[i]);
	i++;
	/* Following statement displays value 40, as i has been
	incremented to 3 and the value at array[3] is 40. */
	printf("\n%d", array[i]);
}
```

4.2 لوپ سٹرکچر (Loop Structure):

اگر ہمیں ایک یا ایک زیادہ سٹیٹمنٹس دُہرانی ہوں تو ہم لوپس کا استعمال کرتے ہیں۔ مثلاً ہم سکرین پر ہزار مرتبہ "Pakistan" لکھنا چاہتے ہیں تو ہم ہزار مرتبہ ;printf("Pakistan") لکھنے کے بجائے لوپ استعمال کریں گے۔ C-لینگویج میں لوپ کی تین قسمیں ہیں:

1- For لوپ

2- While لوپ

3- Do-while لوپ

اس باب میں ہم For لوپ پر غور کریں گے۔

4.2.1 لوپس کا عام ڈھانچہ (General Structure of Loops)

اگر ہم اس پروسیس کا بغور جائزہ لیں جو انسان ایک کام کو ایک خاص حد تک دُہراتے ہوئے استعمال کرتے ہیں تو ہمیں ان لوپس کو سمجھنے میں آسانی ہوگی جو C-لینگویج میں تکرار کو کنٹرول کرنے کے لیے دیے گئے ہیں۔

فرض کریں ہمارے کھیلوں کے استاد نے ہمیں کہا کہ:
ٹریک کے 10 چکر لگائیں۔ ہم یہ کام کس طرح کریں گے؟ پہلے ہم گنتی کو صفر سے شروع کریں گے۔ ہر چکر کے بعد ہم گنتی میں 1 کا اضافہ کرتے ہیں اور دیکھتے ہیں کہ کیا 10 چکر پورے ہو گئے ہیں یا نہیں۔ اگر 10 چکر نہیں پورے ہوئے تو ہم ایک اور چکر لگائیں گے اور پھر گنتی میں 1 کا اضافہ کر کے دیکھیں گے کہ 10 چکر پورے ہوئے یا نہیں۔ ہم یہ کام تب تک دُہرائیں گے جب تک گنتی 10 نہیں ہو جاتی۔

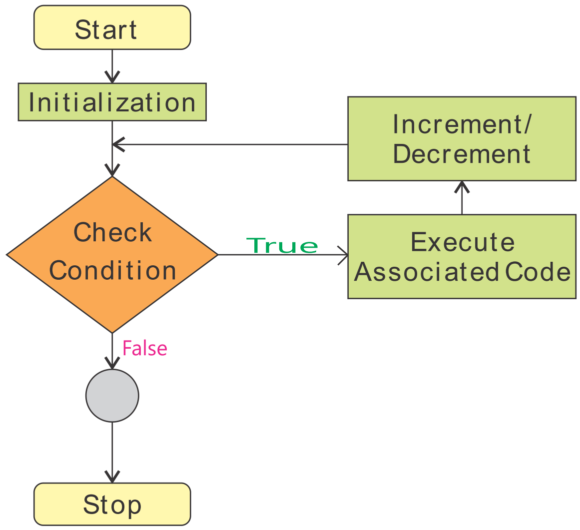
مختلف پروگرامنگ لینگویجز میں ہدایات کے ایک سیٹ کی تکرار کے لیے لوپ سٹرکچرز کا یہی فلسفہ استعمال ہوتا ہے۔

4.2.2 لوپ کا عام سینٹیکس (General Syntax of for loop)

C- پروگرامنگ لینگویج میں FOR لوپ کا عام سینٹیکس یہ ہے:

```
for(initialization; condition; increment/decrement)
{
        Code to repeat
}
```

FOR لوپ کے ڈھانچے کو سمجھنے کے لیے اس فلو چارٹ پر نظر ڈالیں۔



فلو چارٹ میں ہم درج ذیل ترتیب دیکھ سکتے ہیں:

1۔انیشلائزیشن (Initialization) FOR لوپ کا وہ حصہ ہے جو سب سے پہلے ایگزیکیوٹ ہوتا ہے۔اس میں ہم ایک کاؤنٹر ویری ایبل کو انیشلائز کرتے ہیں اور پھر کنڈیشن والے حصے پر جاتے ہیں۔

2۔کنڈیشن (Condition) چیک ہوتی ہے اور اگر یہ غلط (False) ہو تو ہم لوپ سے باہر آجاتے ہیں۔

3۔اگر کنڈیشن درست (True) ہو تو لوپ کی باڈی چلتی ہے۔لوپ کی باڈی ایگزیکیوٹ ہونے کے بعد کاؤنٹر ویری ایبل میں لاجک کے مطابق اضافہ یا کمی ہوتی ہے اور ہم دوبارہ مرحلہ نمبر 2 پر چلے جاتے ہیں۔

EXAMPLE CODE 4.4

```
for(int i = 0; i < 3; i++)
{
      printf("Pakistan\n");
}
```

Output:

```
Pakistan
Pakistan
Pakistan
```

جاری ہے۔

تفصیل:

اگر ہم پچھلے صفحہ پر دیے گئے کوڈ پر غور کریں اور فلو چارٹ سے اس کا موازنہ کریں تو ہم دیکھ سکتے ہیں کہ پروگرام درج ذیل ترتیب سے ایگزیکیوٹ (Execute) ہوتا ہے۔

1۔انیشلائزیشن ایکسپریشن چلتی ہے یعنی ;int i=0 یہاں کا وَنٹرویری ایبل "i" ڈکلیئر ہوجاتا ہے اور قیمت 0 سے انیشلائز ہوجاتا ہے۔

2۔کنڈیشن یعنی i<3 کو ٹیسٹ کیا جاتا ہے۔جیسا کہ i کی قیمت 0 ہے جو 3 سے کم ہے تو کنڈیشن پوری ہوجاتی ہے اور ہم لوپ کی باڈی میں آجاتے ہیں۔

3۔لوپ باڈی چلتی ہے یعنی ;printf("Pakistan\n") اور سکرین پر "Pakistan" لکھا جاتا ہے۔

4۔انکریمینٹ/ڈیکریمینٹ ایکسپریشن ایگزیکیوٹ ہوتی ہے یعنی ;++i چلنے سے i کی قیمت میں 1 کا اضافہ ہوتا ہے۔کیوں کہ i کی قیمت پہلے 0 تھی تو اب 1 ہوجائے گی۔

5۔اب دوبارہ کنڈیشن کو ٹیسٹ کریں گے کیونکہ i کی قیمت 1 ہے جو 3 سے کم ہے تو کنڈیشن پھر پوری ہوجائے گی اور لوپ باڈی چلے گی یعنی سکرین پر دوبارہ "Pakistan" لکھا جائے گا۔پھر i کی قیمت 2 ہوجائے گی۔

6۔اب کنڈیشن دوبارہ چیک کریں گے۔چونکہ i کی قیمت 2 ہے جو 3 سے کم ہے اس لیے سکرین پر دوبارہ "Pakistan" لکھا جائے گا اور i کی قیمت 3 ہوجائے گی۔

7۔شرط دوبارہ چیک ہوگی۔ چوں کہ i کی قیمت 3 ہے جو 3 سے کم نہیں اس لیے کنڈیشن false ہوجائے گی۔اور کنٹرول لوپ سے باہر آجائے گا۔

4.3 پروگرامنگ ٹائم (Programming Time)

ایک پروگرام لکھیں جو 1 سے 10 تک نمبر سکرین پر دکھائے۔

Program:

```
for(int i = 1; i <= 10; i++)
{
      printf("%d\n", i);
}
```

جاری ہے۔

تفصیل:

دیے گئے پروگرام پر غور کریں:۔

1۔سب سے پہلے i کی قیمت 1 رکھی اور کنڈیشن چیک کی۔

2۔جیسا کہ کنڈیشن (10 =>1) ture ہے تو لوپ باڈی چلی۔چونکہ ہم باڈی میں کاؤنٹر کی قیمت دکھا رہے ہیں تو سکرین پر 1 ظاہر ہوا۔

3۔اضافے کے بعد i کی قیمت 2 ہوگئی۔دوبارہ کنڈیشن چیک کی تو وہ ture تھی کیونکہ 10=>2 سے اس لیے 2 پرنٹ ہوا۔

4۔یہ طریقہ کار 10 پرنٹ ہونے تک چلا اور پھر اضافے کے بعد i کی قیمت 11 ہوگئی۔اب کنڈیشن false ہوگئی اس لیے لوپ 1 سے 10 تک نمبر پرنٹ کرنے کے بعد ختم ہوگیا۔

سرگرمی 4.1:

ایک پروگرام لکھیں جو 2 کا ٹیبل پرنٹ کرے۔

اہم نکتہ:

ہمیشہ اس بات کو یقینی بنائیں کہ لوپ کی کنڈیشن آگے کہیں جا کر false ہوجائے ورنہ لوپ چلتا رہے گا اور کبھی بھی ختم نہیں ہوگا۔

کیا آپ جانتے تھے کہ

لوپ کے ایک بار چلنے کو ایک تکرار (Iteration) کہتے ہیں۔

4.4 پروگرامنگ ٹائم (Programming Time)

ایک پروگرام لکھیں جو صارف سے ایک نمبر اِن پُٹ لے اور اس کا فیکٹوریل شمار کرے۔

پروگرام کی منطق اگر ہم ایک مسئلے کو حل کرنا چاہتے ہیں تو سب سے پہلے ہمیں یہ معلوم ہونا چاہیے کہ ہمارا مقصد اصل میں کیا ہے۔
اس مثال میں ہم دیے گئے نمبر کا فیکٹوریل معلوم کرنا چاہتے ہیں تو ہمیں نمبر کے فیکٹوریل کا فارمولہ معلوم ہونا چاہیے۔

$$N! = 1 * 2 * 3 * 4 * ..... * (N-1) * N$$

ہم دیکھ سکتے ہیں کہ کیا پیٹرن دہرایا جارہا ہے لہٰذا ہم لوپ کے ذریعے اسے حل کرسکتے ہیں۔

جاری ہے۔

Program:

```
#include<stdio.h>
void main()
{
     int n, fact = 1;
     printf("Please enter a positive number whose factorial
     you want to find");
     scanf("%d", &n);
     for(int i = 1; i <= n;i ++)
     {
          fact = fact * i;
     }
     printf("The factorial of input number %d is %d", n ,
     fact);
}
```

تفصیل: درج ذیل ٹیبل میں پروگرام کا کام کرنے کا طریقہ دکھایا گیا ہے۔ اگر ان پٹ 5 ہو۔ اس میں ہر تکرار کے بعد ویری ایبلز کو قیمتوں میں تبدیلی واضح کی گئی ہے۔

Iteration	Value of counter	Condition	Loop body	Result
				fact=1
1	i = 1	TRUE (1<=5)	fact = fact * i	fact=1*1=1
2	i = 2	TRUE (2<=5)	fact = fact * i	fact=1*2=2
3	i = 3	TRUE (3<=5)	fact = fact * i	fact=2*3=6
4	i = 4	TRUE (4<=5)	fact = fact * i	fact=6*4=24
5	i = 5	TRUE (5<=5)	fact = fact * i	fact=24*5=120
6	i = 6	FALSE (6>5)		

4.2.3 نیسٹیڈ لوپس (Nested Loops)

آیئے! ایک لوپ کے ڈھانچے کا بغور مشاہدہ کرتے ہیں۔

```
for(initialization; condition; increment/decrement)
{
     Code to repeat
}
```

ہم دیکھ سکتے ہیں کہ دہرایا جانے والا کوڈ (Code to repeat)C-لینگویج کا کوئی بھی کوڈ ہوسکتا ہے یہ ایک اور For لوپ بھی ہوسکتا ہے مثلاً درج ذیل ڈھانچہ ایک درست لوپ سٹرکچر ہے۔جب ہم لوپ کے اندر ایک اور لوپ استعمال کرتے ہیں تو یہ نیسٹیڈ لوپ سٹرکچر(Nested Loop Structure)کہلاتا ہے۔

```
for(initialization; condition; increment/decrement)
{
        for(initialization; condition; increment/decrement)
        {
                Code to repeat
        }
}
```

ہم نیسٹیڈ لوپ کب استعمال کرتے ہیں؟

جب ہم ایک پیٹرن کو ایک سے زیادہ مرتبہ دُہرانا چاہتے ہیں تو ہم نیسٹیڈ لوپس استعمال کرتے ہیں مثلاً اگر ہم دس تک نمبر 10 بار سکرین پر دکھانا چاہتے ہیں تو ہم 1 سے 10 تک نمبر پرنٹ کرنے والے کوڈ کو ایک اور لوپ میں لکھ سکتے ہیں جو دس مرتبہ چلے۔

4.5 پروگرامنگ ٹائم(Programming Time)

پرابلم:

ایک پروگرام لکھیں جو 5 مرتبہ کمپیوٹر کی سکرین پر 1 سے 10 تک نمبر دکھائے۔

پروگرام:

Program:

```
#include<stdio.h>
void main()
{
        for(int i = 1; i <= 5; i++)
        {
                for(int j = 1; j <= 10; j++)
                {
                        printf("%d ", j);
                }
                printf("\n");
        }
}
```

جاری ہے۔

آؤٹ پٹ:

اوپر دیے گئے پروگرام کی آؤٹ پٹ یہ ہے۔

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

تفصیل:

جیسا کہ ہمیں معلوم ہے کہ اندرونی لوپ کس طرح کام کرتا ہے تو یہاں ہم بیرونی لوپ پر غور کریں گے۔

1۔ i کی قیمت 1 ہو تو بیرونی لوپ کی کنڈیشن درست ہو جاتی ہے اس لیے پورا اندرونی لوپ چلتا ہے۔ اور 1 سے 10 تک نمبر پرنٹ کر دیتا ہے۔

2۔ جب کنٹرول اندرونی لوپ سے باہر آتا ہے تو Printf("\n") چلتا ہے اور سکرین پر ایک اور لائن آجاتی ہے۔

3۔ i کی قیمت میں اضافہ ہوتا ہے اور قیمت 2 ہو جاتی ہے۔ جیسا کہ یہ 5 سے کم ہے تو کنڈیشن دوبارہ درست ہو جاتی ہے اور 1 سے 10 تک نمبر سکرین پر پرنٹ ہو جاتے ہیں۔ پھر اندرونی لوپ سے باہر آکر ایک نئی لائن کنسول پر ڈال دی جاتی ہے۔

4۔ 5 مرتبہ 1 سے 10 تک نمبر سکرین پر پرنٹ ہونے کے بعد جب i کی قیمت میں اضافہ ہوتا ہے تو اس کی قیمت 6 ہو جاتی ہے، بیرونی لوپ کی شرط پوری نہیں ہوتی اور بیرونی لوپ ختم ہو جاتا ہے۔

4.6 پروگرامنگ ٹائم (Programming Time)

پرابلم: "*" کا درج ذیل پیٹرن سکرین پر دکھانے کا پروگرام تحریر کریں۔

```
*
**
***
****
*****
******
```

جاری ہے۔

Program:

```
#include<stdio.h>
void main()
{
	for(int i = 1; i <= 6; i++)
	{
		for(int j = 1; j <= i; j++)
			printf("*");
		printf("\n");
	}
}
```

تفصیل: درج بالا کوڈ کی تفصیل یہ ہے:-

1۔ جیسا کہ ہم ستاروں کی چھ لائنیں پرنٹ کرنا چاہتے ہیں اس لیے باہر والا لوپ 1 سے 6 تک چلے گا۔

2۔ ہم دیکھ سکتے ہیں کہ دیے گئے پیٹرن میں پہلی لائن پر ایک ستارہ ہے دوسری پر دو، تیسری پر تین اور باقی بھی ایسے ہی ہیں۔ اس لیے اندرونی لوپ بیرونی لوپ پر منحصر ہے یعنی اگر بیرونی لوپ کا کاؤنٹر 1 ہے تو اندرونی لوپ ایک دفعہ چلے گا، اگر کاؤنٹر 2 ہے تو اندرونی لوپ 2 دفعہ چلے گا اور اسی طرح باقی۔ اس لیے ہم بیرونی لوپ کا کاؤنٹر اندرونی لوپ کی کنڈیشن میں استعمال کرتے ہیں یعنی j<=i

3۔ جب بیرونی لوپ کا کاؤنٹر کی قیمت 1 ہوتی ہے تو اندرونی لوپ ایک بار چلتا ہے اس لیے صرف ایک ستارہ پرنٹ ہوتا ہے۔ جب بیرونی لوپ کا کاؤنٹر 2 ہوتا ہے تو اندرونی لوپ دوبارہ چلتا ہے اور دو ستارے پرنٹ ہوتے ہیں۔ یہ پروسیس اسی طرح دُہرایا جاتا ہے۔ جب تک چھے سطریں مکمل نہیں ہو جاتیں۔

سرگرمی 4.2:

ایک پروگرام لکھیں جو 2، 3، 4، 5 اور 6 کے ٹیبل پرنٹ کرے۔

اہم نکتہ:

ہم if سٹرکچر کو لوپ سٹرکچر میں اور لوپ سٹرکچر کو if سٹرکچر میں کسی بھی قابلِ ذکر طریقے سے استعمال کر سکتے ہیں۔

4.2.4 حل شدہ مثالیں

4.7 پروگرامنگ ٹائم (Programming Time)

پرابلم:

ایک پروگرام لکھیں جو بتائے کہ دو نمبروں کے درمیان دیے گئے نمبر کے کتنے حاصل ضرب آتے ہیں؟

Program:

```
#include <stdio.h>
void main ()
{
      int n, lower, upper, count = 0;
      printf ("Enter the number: ");
      scanf ("%d", &n);
      printf  ("Enter  the  lower  and  upper  limit  of
      multiples:\n");
      scanf ("%d%d", &lower, &upper);
      for(int i = lower; i <= upper; i++)
            if(i % n == 0)
                  count++;
      printf ("Number of multiples of %d between %d and %d are
      %d", n, lower, upper, count);
}
```

4.8 پروگرامنگ ٹائم (Programming Time)

پرابلم:

ایک پروگرام لکھیں جو n1 سے n2 تک آنے والے انٹیجرز (integers) میں سے جفت نمبر بتائے (n1 بڑا ہے n2 سے)

Program:

```
#include <stdio.h>
void main ()
{
      int n1, n2;
      printf  ("Enter  the  lower  and  upper  limit  of  even
      numbers:\n");
      scanf ("%d%d", &n2, &n1);
```

جاری ہے۔

```
            if(n1 > n2)
            {
                  for (int i = n1; i >= n2; I--)
                  {
                        if(i % 2 == 0)
                              printf ("%d ", i);
                  }
            }
      }
```

4.9 پروگرامنگ ٹائم (Programming Time)

پرابلم:

ایک پروگرام لکھیں جو معلوم کرے کہ دیا گیا نمبر مفرد ہے یا نہیں؟

Program:

```
#include <stdio.h>
void main ()
{
      int n;
      int flag = 1;
      printf ("Enter a number: ");
      scanf ("%d", &n);
      for (int i = 2; i < n; i++)
      {
            if (n % i == 0)
                  flag = 0;
      }
      if (flag == 1)
            printf ("This is a prime number");
      else
            printf ("This is not a prime number");
}
```

4.10 پروگرامنگ ٹائم (Programming Time)

پرابلم:

ایک پروگرام لکھیں جو 2 سے 100 کے درمیان پائے جانے والے مفرد اعداد سکرین پر دکھائے۔

Program:

```
# include<stdio.h>
int main ()
{
	int flag;
	for (int j = 2; j <= 100; j++)
	{
		flag = 1;
		for (int i = 2; i < j; i++)
		{
			if(j % i == 0)
			{
				flag = 0;
			}
		}
		if (flag == 1)
		{
			printf ("%d ", j);
		}
	}
}
```

4.2.5 لوپس اور اریز (Loops and Arays)

چوں کہ ویری ایبلز کو ارے انڈیکسز کے طور پر استعمال کیا جا سکتا ہے اس لئے ہم اریز پر مختلف آپریشنز انجام دینے کے لیے لوپ کا استعمال کر سکتے ہیں۔ اگر ہم پوری ارے پرنٹ کرنا چاہتے ہیں تو بجائے ایک ایک کر کے تمام ایلیمنٹس کو لکھنے کے ہم لوپ کا وئنٹر کو بطور ارے انڈیکس استعمال کر کے ارے پر لوپ چلا سکتے ہیں۔

اب ہم یہ دیکھیں گے کہ ارے میں قیمتیں لکھنے اور پڑھنے کے لیے لوپ کا استعمال کس طرح کیا جا سکتا ہے۔

1۔لوپ کو استعمال کر کے ارے میں قیمتیں لکھنا

لوپس کو استعمال کر کے ہم ارے میں بآسانی ان پٹ لے سکتے ہیں اگر ہم سائز 10 کی ارے میں صارف سے ان پٹ لینا چاہتے ہیں تو ہم اس طرح سے لوپ استعمال کر سکتے ہیں۔

EXAMPLE CODE 4.5

```
int a[10];
for (int i = 0; i < 10; i++)
    scanf ("%d", &a[i]);
```

4.11 پروگرامنگ ٹائم (Programming Time)

پرابلم:

ایک پروگرام لکھیں جو سائز 5 کی ایک ارے میں 23 کے پہلے 5 حاصل ضرب لکھے۔

Program:

```
#include<stdio.h>
void main()
{
    int multiples[5];
    for (int i = 0; i < 5; i++)
        multiples[i]= (i + 1) * 23 ;
}
```

2۔لوپ کو استعمال کر کے ارے سے قیمتیں پڑھنا

اب ہم دیکھتے ہیں کہ ارے سے قیمتیں پڑھنے میں لوپ ہماری کس طرح مدد کرتا ہے درج ذیل کوڈ کو استعمال کر کے ہم 100 ایلمنٹس کی ایک ارے سکرین پر دکھا سکتے ہیں۔

EXAMPLE CODE 4.6

```
for (int i = 0; i < 100; i++)
    printf("%d ", a[i]);
```

درج ذیل کوڈ کو استعمال کرتے ہوئے ہم 100 ایلیمنٹس کی ایک ارے کے تمام ایلیمنٹس کو جمع کر سکتے ہیں۔

EXAMPLE CODE 4.7

```
int sum = 0;
for(int i = 0; i < 100; i++)
      sum = sum + a[i];
printf("The sum of all the elements of array is %d", sum);
```

سرگرمی 4.3:

ایک پروگرام لکھیں جو ایک جماعت کے 30 طلبہ کے میٹرک کے نمبر ان پٹ لے اور ان کی اوسط بتائے۔

4.2.6 حل شدہ مثالیں

4.12 پروگرامنگ ٹائم (Programming Time)

پرابلم:

ایک پروگرام لکھیں جو دو اریز کے متعلقہ ایلیمنٹس کو جمع کرے۔

Program:

```
#include <stdio.h>
void main ()
{
      int a[] = {2, 3, 54, 22, 67, 34, 29, 19};
      int b[] = {65,73,26, 10, 4, 2, 84, 26};
      for (int i=0; i<8; i++)
            printf ("%d   ", a[i] + b[i]);
}
```

خلاصہ:

- ڈیٹا سٹرکچر ایک کنٹینر ہوتا ہے جو ڈیٹا آئٹمز (Data Items) کے مجموعے کو ایک خاص ترتیب میں محفوظ کرنے کے لیے استعمال ہوتا ہے۔
- **Array** ارے ایک ڈیٹا سٹرکچر ہے جس میں ایک ڈیٹا ٹائپ کی ایک سے زیادہ قیمتیں لکھی جا سکتی ہیں۔ یہ تمام قیمتیں کمپیوٹر میموری میں منسلک مقامات پر محفوظ کرتا ہے۔
- C- لینگویج میں ارے کو اس طرح ڈکلیئر کرتے ہیں:

data_type array_name[array_size];

- ڈیٹا ٹائپ (Data Type) اس ڈیٹا کی ٹائپ ہے جو ارے میں محفوظ کرتا ہے۔
- ارے کا نام (Array Name) ایک منفرد شناخت ہے جو ارے کا حوالہ دیتا ہے۔
- ارے کا سائز (Array Size) بتاتا ہے کہ زیادہ سے زیادہ کتنے ایلیمنٹس ایک ارے میں رکھے جا سکتے ہیں۔
- ارے میں پہلی مرتبہ قیمتیں لکھنا، ارے انشلائزیشن کہلاتا ہے۔ ایک ارے کو ڈکلیئریشن کے وقت یا بعد میں انشلائز کیا جا سکتا ہے۔

ڈکلیئریشن کے ساتھ ہی ارے کو انشلائز کرنے کا طریقہ ہے۔

data_type array_name[N] = {value1, value2, value3,..., valueN};

- ارے کے ہر ایلیمنٹ کے ایک انڈیکس ارے کے نام کے ساتھ اس طرح Array Name[Index] لکھ کر اس انڈیکس پر محفوظ ڈیٹا تک رسائی حاصل کی جا سکتی ہے۔ متغیرات کو بھی ارے انڈیکس کے طور پر استعمال کیا جا سکتا ہے۔
- لوپ سٹرکچر سٹیٹمینٹس کے ایک سیٹ کو دُہرانے کے لیے استعمال ہوتا ہے۔ لوپ کی تین قسمیں For لوپ، While لوپ، do while لوپ ہیں۔
- C- پروگرمنگ لینگویج میں For لوپ کا عام ڈھانچہ ہے۔

```
for(initialization; condition; increment/decrement)
{
        Code to repeat;
}
```

- جب ہم لوپ کے اندر ایک اور لوپ استعمال کرتے ہیں تو یہ نیسٹیڈ لوپ سٹرکچر کہلاتا ہے۔ نیسٹیڈ لوپس ایک پیٹرن کو بار بار دُہرانے کے لیے استعمال ہوتے ہیں۔
- لوپس کے ذریعے اریز میں قیمتیں لکھنا اور پڑھنا آسان ہو جاتا ہے۔

مشق

سوال نمبر 1 کثیر الانتخابی سوالات:

1۔ارے ایک ۔۔۔۔۔۔۔۔۔۔۔سٹرکچر ہے:

a) لوپ b) کنٹرول c) ڈیٹا d) مشروط

2۔ارے کے ایلیمنٹس میموری کے مقامات ۔۔۔۔۔۔۔۔۔۔پر محفوظ ہوتے ہیں:

a) منسلک b) بکھرے ہوئے c) تقسیم شدہ d) کوئی بھی نہیں

3۔اگر ارے کا سائز 100 ہے تو انڈیکسز کی رینج ۔۔۔۔۔۔۔۔۔۔۔ہوگی:

a) 0-99 b) 0-100 c) 1-100 d) 2-2012

4۔۔۔۔۔۔۔۔سٹرکچر ہمیشہ ہدایات کے مجموعے کو بار بار دہرانے کے لیے استعمال ہوتا ہے:

a) لوپ b) مشروط c) کنٹرول d) ڈیٹا

5۔۔۔۔۔۔۔۔۔ایک مخصوص شناخت ہے جو ارے کا حوالہ دیتا ہے:

a) ڈیٹا ٹائپ b) ارے کا نام c) ارے کا سائز d) کوئی بھی نہیں

6۔ارے کو ڈکلیئریشن کے ۔۔۔۔۔۔۔۔انیشلائز کیا جاسکتا ہے:

a) اس وقت b) اس کے بعد c) اس کے پہلو d) (a) اور (b) دونوں

7۔لوپس کے اندر لوپس کا استعمال ۔۔۔۔۔۔۔لوپس کہلاتا ہے:

a) for b) while c) do while d) نیسٹیڈ

8۔For لوپ کا ۔۔۔۔۔۔۔۔۔۔۔حصہ سب سے پہلے چلتا ہے:

a) شرط b) باڈی c) انیشلائزیشن d) اضافہ/کمی

9۔۔۔۔۔۔۔۔۔سے ارے میں قیمتیں لکھنا اور پڑھنا آسان ہو جاتا ہے:

a) لوپس b) شرائط c) ایکسپریشنز d) فنکشنز

10۔ارے کو ایک سٹیٹمنٹ میں انیشلائز کرنے کے لئے اسے ڈکلیئریشن کے ۔۔۔۔۔۔۔۔۔۔۔انیشلائز کریں:

a) وقت b) بعد c) پہلے d) (a) اور (b) دونوں

سوال نمبر 2: درج ذیل اِصطلاحات کی تعریف کریں۔

1۔ڈیٹا سٹرکچر　2۔ارے　3۔ارے انیشلائزیشن　4۔لوپ سٹرکچر　5۔نیسٹیڈ لوپس

سوال نمبر 3: درج ذیل سوالات کے مختصر جوابات دیں۔

1۔ کیا لوپ ایک ڈیٹا سٹرکچر ہے؟ اپنے جواب کی توثیق کریں۔

2۔ نیسٹیڈ لوپس کا استعمال کیا ہے؟

3۔ ایک ارے کو ڈکلیئریشن کے وقت انیشلائز کرنے کا کا فائدہ کیا ہے؟

4۔for لوپ کے ڈھانچے کی وضاحت کریں۔

5۔ آپ ارے کو کیسے ڈکلیئر کر سکتے ہیں؟ ارے ڈکلیئریشن کے تین حصوں کی مختصراً وضاحت کریں۔

سوال نمبر 4: کوڈ کے درج ذیل حصوں میں ایررز تلاش کریں۔

a)
```
int a[] = ({2},{3},{4});
```

b)
```
for (int i = 0, i < 10, i++)
      printf (“%d\n”, i);
```

c)
```
int a[] = {1,2,3,4,5};
for (int j = 0; j < 5; j++)
      printf (“%d “, a(j));
```

d)
```
float f[] = {1.4, 3.5, 7.3, 5.9};
int size = 4;
for (int n = -1; n < size; n--)
      printf (“%f\n”, f[n]);
```

e)
```
int count = 0;
for (int i = 4; i < 6; i--)
      for (int j = i, j < 45; j++)
      {
            count++;
            printf (“%count”, count)
      }
```

سوال نمبر 5: کوڈ کے درج ذیل حصوں کی آؤٹ پٹ لکھیں۔

a)
```
int sum = 0, p;
for (p = 5; p <= 25; p = p + 5)
      sum = sum + 5;
printf ("Sum is %d", sum);
```

b)
```
int i;
for (i = 34; i <= 60; i = i * 2)
      printf ("* ");
```

c)
```
for (int i = 50; i <= 50; i++)
{
      for (j = i; j >= 48; j--)
            printf ("j = %d \ n", j);
      printf ("i = %d\n", i);
}
```

d)
```
int i, arr[] = {2, 3, 4, 5, 6, 7, 8};
for (i = 0; i < 7; i++)
{
      printf ("%d\n", arr[i] * arr[i]);
      i++;
}
```

e)
```
int i,j;
float ar1[] = {1.1, 1.2, 1.3};
float ar2[] = {2.1, 2.2, 2.3};
for (i = 0; i < 3; i++)
      for (j = i; j < 3; j++)
            printf ("%f\n", ar1[i] * ar2[j] * i * j);
```

پروگرامنگ کی مشقیں

مشق 1:

لوپس کو استعمال کر کے کنسول پر پیٹرن پرنٹ کریں۔

a)
```
*****
*****
*****
```

b)
```
A
BC
DEF
GHIJ
KLMN
```

مشق 2:

ایک پروگرام لکھیں جو دو مثبت نمبر a اور b ان پٹ لے اور a^b شمار کر کے سکرین پر دکھائیں۔

مشق 3:

ایک پروگرام لکھیں جو 2 نمبر ان پٹ لے اور یوکلیڈین میتھڈ (Euclidean Method) کو استعمال کرتے ہوئے GCD معلوم کر کے بتائے۔

مشق 4:

ایک پروگرام لکھیں جو 1 سے 7 تک نمبرز کا فیکٹوریل سکرین پر دکھائے (ہینٹ: نیسٹڈ لوپ کو استعمال کریں)

مشق 5:

ایک پروگرام لکھیں جو دس ایلیمنٹس کی ایک ارے کو ڈکلیئر اور انیشلائز کرے اور پہلے اور آخری ایلیمنٹ کو ضرب کر کے جواب سکرین پر دکھائے۔

مشق 6:

ایک پروگرام لکھیں جو 7 ایلیمنٹس کی ارے ڈکلیئر اور انیشلائز کرے اور یہ بتائے کہ ارے میں کتنے ایلیمنٹس 10 سے بڑھے ہیں۔