

## فنکشنز

تدریسی مقاصد: (Student Learning Outcomes)

- فنکشنز کا تصور اور ان کی اقسام کی وضاحت
- فنکشنز کے استعمال کے فوائد کی وضاحت
- فنکشنز کے سگنچر (نام، آرگومینٹس، ریٹرن ٹائپ) کی وضاحت
- فنکشنز سے متعلق اصطلاحات کی وضاحت۔
- فنکشن کی تعریف
- فنکشن کا استعمال
- C لینگویج کو استعمال کرتے ہوئے درج ذیل فنکشنز لکھنا:
  - ایک فنکشن جو دو انٹیجر (integer) وییری ایبلز بطور آرگومینٹ لے اور ان کی حاصل واپس کرے۔
  - ایک فنکشن جو تین متغیرات لے اور ان کا درمیان والا نمبر واپس کرے۔



**یونٹ کا تعارف (Unit Introduction)**

کسی بھی مسئلے کو حل کرنے کی اچھی حکمت عملی یہ ہے کہ اسے چھوٹے چھوٹے حصوں میں تقسیم کر دیا جائے۔ ایک ایک حصے کا حل نکال کر ان سب کو اکٹھا کرنے سے پورے مسئلے کا حل مل جاتا ہے۔ سارا وقت پورے مسئلے کے بارے میں سوچنے کے بجائے ایک وقت میں ایک چھوٹے حصے پر غور کرنا آسان ہوتا ہے۔ مسئلے کا حل نکالنے کی اس حکمت عملی کو تقسیم کرنا اور فتح کرنا (Divide and conquer) کہتے ہیں۔ C پر وگرامنگ لینگویج میں ہمارے پاس فنکشنز ہوتے ہیں جو پروگرامنگ کے سوال کو حل کرنے کے لیے تقسیم کرنے اور فتح کرنے کی حکمت عملی استعمال کرتے ہیں۔ اس باب میں ہم فنکشنز کا تصور ان کے فوائد اور ان کے ساتھ کام کرنے کا طریقہ سیکھیں گے۔

**5.1 فنکشنز (Functions)**

فنکشن سٹیٹمنٹس کا ایک بلاک ہے جو ایک خاص کام انجام دیتا ہے مثلاً printf ایک فنکشن ہے جو کمپیوٹر کی سکرین پر کچھ بھی دکھانے کے لیے استعمال ہوتا ہے۔ scanf ایک فنکشن ہے جو صارف سے ان پٹ لینے کے لیے استعمال ہوتا ہے۔ ہر پروگرام میں ایک main فنکشن ہوتا ہے۔ جو صارف کے مطلوب افعال پروگرام کو سرانجام دیتا ہے۔ اسی طرح ہم اور فنکشنز بھی لکھ سکتے ہیں۔ اور انہیں کئی مرتبہ استعمال کر سکتے ہیں۔

**5.1.1 فنکشنز کی اقسام**

بنیادی طور پر فنکشنز کی دو اقسام ہیں:

- 1- بلٹ ان فنکشنز (Built-in Function)
- 2- یوزر ڈیفائنڈ فنکشنز (User-Defined Function)

**بلٹ ان فنکشنز (Built in Function)**

وہ فنکشنز جو C کی سٹینڈرڈ لائبریری میں موجود ہیں بلٹ ان فنکشنز کہلاتے ہیں۔ یہ فنکشنز عام طور پر ریاضی کے حساب کتاب، سٹرنگ آپریشنز، ان پٹ اور آؤٹ پٹ آپریشنز وغیرہ انجام دیتے ہیں مثلاً printf اور scanf بلٹ ان فنکشنز ہیں۔

**یوزر ڈیفائنڈ فنکشنز (User-defined Function)**

وہ فنکشنز جو پروگرامر ڈیفائن کرتا ہے یوزر ڈیفائنڈ فنکشنز کہلاتے ہیں اس باب میں ہم یوزر ڈیفائنڈ فنکشنز لکھنا سیکھیں گے۔

## 5.1.2 فنکشنز کے فوائد

فنکشنز سے ہمیں درج ذیل فوائد حاصل ہوتے ہیں :-

## 1- دوبارہ استعمال (Reusability)

فنکشنز کے ذریعے ہم کوڈ کو دوبارہ استعمال کر سکتے ہیں۔ اس کا مطلب یہ ہے کہ جب بھی ہمیں ایک فنکشن کی فنکشنیلٹی (Functionality) کی ضرورت ہو ہم اس فنکشن کو کال (call) کر سکتے ہیں۔ ہمیں سٹیٹمنٹس کا ایک سیٹ بار بار نہیں لکھنا پڑتا۔

## 2- کاموں کو الگ کرنا (Separation of Tasks)

فنکشن کے ذریعے ہم ایک کام کرنے کے کوڈ کو دوسرے کام کے کوڈ سے الگ کر سکتے ہیں۔ اگر ہمیں ایک فنکشن میں مسئلہ ہو تو اسے حل کرنے کے لیے پورے پروگرام کو چیک نہیں کرنا پڑتا۔ ہمیں صرف ایک فنکشن پر غور کرنا ہوتا ہے۔

## 3- مسئلے کی پیچیدگی سے نمٹنا (Handling the Complexity of Problem)

اگر ہم پورا پروگرام ایک پروسیجر کے طور پر لکھیں تو اس کی دیکھ بھال (manage) کرنا مشکل ہو جاتا ہے۔ فنکشنز کے ذریعے ہم پروگرام کو چھوٹے حصوں میں تقسیم کرتے ہیں اس سے مسئلے کی پیچیدگی کم ہو جاتی ہے۔

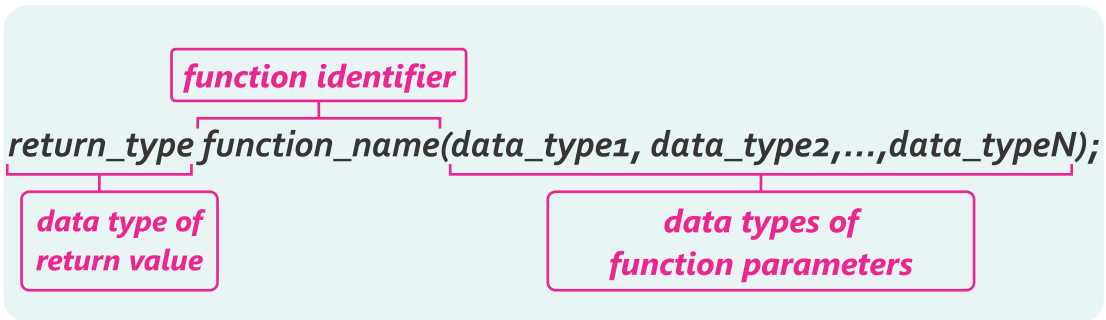
## 4- پڑھنے کی صلاحیت (Readability)

پروگرام کو کئی فنکشنز میں تقسیم کرنے سے اس کو پڑھنا (سمجھنا) زیادہ آسان ہو جاتا ہے۔

## 5.1.3 فنکشن کا سگنیچر (Signature of Function):

فنکشن سٹیٹمنٹ کا ایک بلاک ہوتا ہے جو کچھ ان پٹ لیتا ہے۔ فنکشن کی ان پٹس کو پیرامیٹرز کہتے ہیں اور آؤٹ پٹ کو ریٹرن ویلو (Return Value) کہتے ہیں۔ ایک فنکشن کے ایک سے زیادہ پیرامیٹرز تو ہو سکتے ہیں لیکن وہ ایک ہی قیمت ریٹرن کر سکتا ہے زیادہ نہیں۔

فنکشن سگنیچر فنکشن کی ان پٹس اور آؤٹ پٹ ڈیفائن کرتا ہے۔ فنکشن کے سگنیچر کا عام ڈھانچہ یہ ہے۔



## فنکشن سگنچرز کی مثالیں:

ٹیبل 5.1 میں کچھ فنکشنز اور ان کے سگنچرز کی تفصیل دی گئی ہے:-

فنکشن سگنچر	فنکشن کی تفصیل
<code>int square (int);</code>	ایک فنکشن جو ایک انٹیجر (int) ان پٹ لیتا ہے۔ اور اس کا مربع واپس کرتا ہے۔
<code>float perimeter (float, float);</code>	ایک فنکشن جو مستطیل کی لمبائی اور چوڑائی ان پٹ لے اور اس کا احاطہ ریٹرن کرے۔
<code>int largest (int, int, int);</code>	ایک فنکشن جو تین انٹیجر (integers) ان پٹ لے اور ان میں سے سب سے بڑی ویلیو ریٹرن کرے۔
<code>float area (float);</code>	ایک فنکشن جو دائرے کا رداس ان پٹ لے اور رقبہ ریٹرن کرے۔
<code>int isVowel (char);</code>	ایک فنکشن جو ایک کریکٹر ان پٹ لے اور اگر وہ حرفِ علت ہو تو 1 ریٹرن کرے ورنہ 0 ریٹرن کرے۔

ٹیبل 5.1 کچھ فنکشنز اور ان کے سگنچرز

## 5.1.4 فنکشن کو ڈیفائن کرنا (Defining a function)

ایک فنکشن کے سگنچر سے یہ نہیں پتا چلتا کہ وہ کس کام کے لیے ہے۔ اس کے لیے فنکشن ڈیفینیشن (Function Definition) ہوتی ہے۔ ایک فنکشن ڈیفینیشن کا ڈھانچہ کچھ اس طرح سے ہے:

**`return_type function_name (data_type var1, data_type var2,..., data_type varN)`**

{

**Body of the function**

}

فنکشن کی باڈی ان سٹیٹمنٹس کا سیٹ ہوتا ہے جنہیں چلا کر فنکشن کوئی خاص کام سرانجام دیتا ہے۔ فنکشن سگنچر کے فوراً بعد { } تو سین میں لکھی گئی سٹیٹمنٹس فنکشن کی باڈی بناتی ہیں۔ درج ذیل مثال ایک فنکشن کو ڈیفائن کرتی ہے (showPangram) جو نہ تو کچھ ان پٹ لیتا ہے نہ کچھ ریٹرن کرتا ہے بس کمپیوٹر کی سکرین پر "A Quick Brown Fox Jumps Over the Lazy Dog" دکھاتا ہے۔

## EXAMPLE CODE 5.1

```
void showPangram()
{
    printf("\nA quick brown fox jumps over the lazy
    dog.\n");
}
```

چوں کہ درج بالا فنکشن کچھ ریٹرن نہیں کرتا اس لیے اس کی ریٹرن ٹائپ void ہے۔  
آئیے! اب ایک اور مثال لیتے ہیں جو دو انٹیجر (integer) ان پٹ لے اور ان کی جمع ریٹرن کرے۔

## EXAMPLE CODE 5.2

```
int add(int x, int y)
{
    int result;
    result = x + y;
    return result;
}
```

فنکشن کے اندر return ایک مطلوبہ لفظ ہے جو کالنگ (calling) فنکشن کو قیمت ریٹرن کرنے کے لیے استعمال ہوتا ہے۔

## اہم نوٹ:

ایک فنکشن ایک سے زیادہ قیمتیں ریٹرن نہیں کر سکتا مثلاً:  
درج ذیل سٹیٹمنٹ کے نتیجے میں کمپائلر ایرر دیتا ہے۔

```
return (4,5);
```

**اہم نوٹ:**

ایک فنکشن میں ایک سے زیادہ ریٹرن سٹیٹمنٹس ہو سکتی ہیں لیکن جیسے ہی ایک ریٹرن سٹیٹمنٹ چلتی ہے فنکشن کی کال ختم ہو جاتی ہے۔ فنکشن کی باڈی کی باقی سٹیٹمنٹس نہیں چلتیں۔

**فنکشن کا استعمال:**

ہمیں ایک فنکشن کو کال کرنا پڑتا ہے تاکہ وہ پروگرام کو سونپا گیا کام انجام دے فنکشن کال کے لیے یہ ڈھانچہ استعمال ہوتا ہے۔

**function\_name(value1, value2,..., valueN);**

مثال کے طور پر درج ذیل پروگرام کو دیکھیں۔

**EXAMPLE CODE 5.3**

```
void main()
{
    printf("Hello from main()");
    showPangram(); ← function call
    printf("Welcome back to main()");
}
```

**Output:**

```
Hello from main()
A quick brown fox jumps over the lazy dog.
Welcome back to main()
```

ہم دیکھ سکتے ہیں کہ ایک پروگرام main فنکشن سے چلنا شروع کرتا ہے۔ جب کوئی فنکشن کال (مستطیل کے اندر) آتی ہے تو کنٹرول اس فنکشن کو ٹرانسفر ہو جاتا ہے۔ کال کیے گئے فنکشن کے ایگزیکیوٹ ہونے کے بعد کنٹرول واپس اس فنکشن کے پاس چلا جاتا ہے جس نے فنکشن کال کیا ہوتا ہے جیسے اوپر والی مثال میں main()۔

درج ذیل پروگرام 2 نمبر ان پٹ کے طور پر لیتا ہے اور ان کا مجموعہ ریٹرن کرتا ہے۔

درج ذیل کوڈ میں مستطیل کے اندر جو سٹیٹمنٹ ہے وہ پچھلے سیکشن میں ڈیفائن کیے ہوئے فنکشن "add" کو کال کرتی ہے۔

## EXAMPLE CODE 5.4

```

void main ()
{
    int n1, n2, sum;
    scanf ("%d%d", &n1, &n2);
    sum = add (n1, n2);
    printf ("Sum is %d", sum);
}

```

function name

function call

function arguments

فونکشن کال میں n1 اور n2 فونکشن add() کے آرگومینٹس ہیں۔

فونکشن add() سے ریٹرن ہونے والے نتیجے کو محفوظ کرنے کے لیے متغیر sum ڈیکلیر کیا گیا ہے۔

فونکشن کو بطور آرگومینٹ پاس کیے گئے ویری ایبلز میں کوئی تبدیلی نہیں آتی۔ فونکشن ان ویری ایبلز کی ایک کاپی بناتا ہے اور صرف اس کاپی

میں تبدیلیاں کرتے ہیں۔

درج بالا مثال میں جب n1 اور n2 پاس کیے جاتے ہیں تو فونکشن ان ویری ایبلز کی کاپیاں بنالیتا ہے۔ ویری ایبل n1 کی کاپی x ہے اور

ویری ایبل n2 کی کاپی y ہے۔

## اہم نوٹ:

جو قیمتیں فونکشن کو پاس کی جاتی ہیں وہ آرگومینٹس کہلاتی ہیں جب کہ فونکشن ڈیفینیشن میں جن ویری ایبلز میں یہ قیمتیں جاتی ہیں وہ

فونکشن کے پیرامیٹرز کہلاتے ہیں۔

اوپر دی گئی مثال میں ویری ایبلز n1 اور n2 آرگومینٹس ہیں جو فونکشن add() کو پاس

کیے گئے ہیں جبکہ فونکشن add() کے اندر ویری ایبلز x اور y اس کے پیرامیٹرز ہیں

**اہم نوٹ:**

ضروری نہیں کہ فنکشن کو جو ویری ایبلز پاس کیے جائیں ان کے نام وہی ہوں جو فنکشن کے پیرامیٹرز کے نام ہوں۔ البتہ ہم ایک جیسے نام بھی استعمال کر سکتے ہیں یہاں ایک اہم نکتہ یہ ہے کہ اگر ہم ایک جیسے نام استعمال کریں گے تو بھی فنکشن میں استعمال ہونے والے ویری ایبلز اصل ویری ایبلز کی کاپی ہوں گے۔ یہ درج ذیل مثال سے واضح کیا گیا ہے۔

**EXAMPLE CODE 5.5**

```
#include<stdio.h>

void fun(int x, int y)
{
    x = 20;
    y = 10;
    printf("Values of x and y in fun(): %d %d", x, y);
}

void main()
{
    int x = 10, y = 20;
    fun(x, y);
    printf("Values of x and y in main(): %d %d", x, y);
}
```

**Output:**

Values of x and y in fun(): 20 10

Values of x and y in main(): 10 20



**اہم نوٹ:**

پروگرام میں فنکشنز کو ترتیب دیتے ہوئے درج ذیل نکات ذہن میں رکھیں۔

1- اگر کال کیے گئے فنکشن کی ڈیفینیشن کال کرنے والے فنکشن کی ڈیفینیشن سے پہلے آئے تو فنکشن سگنچر کال کرنے والے فنکشن کی ڈیفینیشن سے پہلے لکھنا ضروری ہے۔

2- اگر کال کیے گئے فنکشن کی ڈیفینیشن کال کرنے والے فنکشن کے بعد میں آئے تو کال کیے گئے فنکشن کا سگنچر اس کال کرنے والے فنکشن سے پہلے لکھنا ضروری ہے۔

نیچے دیے گئے دونوں کوڈسٹرکچرز درست ہیں۔

```
a) int add(int, int);
void main()
{
    printf("%d "add(4, 5));
}
int add(int a, int b)
{
    return a + b;
}
```

```
b) int add(int a, int b)
{
    return a + b;
}
void main()
{
    printf("%d "add(4, 5)
}
```

**5.1 پروگرامنگ ٹائم (Programming Time)**

ایک فنکشن prime() لکھیں جو ایک نمبر ان پٹ لے اور 1 ریٹرن کرے اگر نمبر مفرد ہو تو نہیں تو 0 ریٹرن کرے۔ اس فنکشن کو main() میں استعمال کریں۔

**Program:**

```
#include <stdio.h>
int prime (int n)
{
    for (int i = 2; i < n; i++)
        if(n % i == 0)
            return 0;
    return 1;
}
```

جاری ہے۔

```

void main()
{
    int x;
    printf ("Please enter a number: ");
    scanf ("%d", &x);
    if(prime(x))
        printf ("%d is a Prime Number", x);
    else
        printf ("%d is not a Prime Number", x);
}

```

## 5.2 پروگرامنگ ٹائم (Programming Time)



پراہم: ایک فنکشن لکھیں جو ایک مثبت نمبر ان پٹ کے طور پر لے اور 0 سے لے کر اُس نمبر تک نمبروں کو جمع کرے اور حاصل جمع ریٹرن کرے۔

**Program:**

```

int digitsSum(int n)
{
    int sum = 0;
    for(int i = 0; i <= n; i++)
    {
        sum = sum + i;
    }
    return sum;
}

void main()
{
    int number;
    printf("Please enter a positive number: ");
    scanf("%d", &number);
}

```

جاری ہے۔

```
if(number >= 0)
{
    int sum = digitsSum(number);
    printf("The sum of numbers upto given number is
%d", sum);
}
else
    printf("You entered a negative number.");
}
```

## خلاصہ

- فنکشن سٹیٹمنٹس کا ایک بلاک ہے جو ایک خاص کام انجام دیتا ہے۔
- C- سٹیٹڈ رڈ لائبریری میں موجود فنکشنز بلٹ ان فنکشنز کہلاتے ہیں۔
- پروگرامر جو فنکشنز ڈیفائن کرتا ہے وہ یوزر ڈیفائنڈ فنکشنز کہلاتے ہیں۔
- فنکشن استعمال کرنے کے کچھ فوائد یہ ہیں: کوڈ کو دوبارہ استعمال کرنا، کاموں کو الگ کرنا، مسئلے کی پیچیدگی کو کم کرنا اور کوڈ کو پڑھنے کے قابل بنانا۔
- فنکشن سیگنچر فنکشن کے نام، ان پٹس اور آؤٹ پٹ کی وضاحت کرتا ہے۔
- فنکشن کو اس طرح ڈیفائن کیا جاسکتا ہے۔

```
return_type name (Parameters)
```

```
{
```

```
Body of the Function
```

```
}
```

- فنکشن کی ریٹرن ٹائپ اس قیمت کی ڈیٹا ٹائپ ہوتی ہے جو فنکشن ریٹرن کرتا ہے۔
- فنکشن کا نام اس کے کام سے متعلق ہونا چاہیے۔
- پیرامیٹرز مختلف ڈیٹا ٹائپس کے متغیرات ہوتے ہیں جن میں فنکشن کو بطور ان پٹ پاس کی گئی قیمتیں رکھی جاتی ہیں۔
- فنکشن کی باڈی ان سٹیٹمنٹس کا سیٹ ہوتی ہے جنہیں چلا کر فنکشن مخصوص کام سرانجام دیتا ہے۔
- ایک فنکشن کو کال کرنے سے مراد اس فنکشن کو کنٹرول ٹرانسفر کرنا ہے۔
- فنکشن کال کے دوران جو قیمتیں فنکشن کو پاس کی جاتی ہیں وہ آرگیومنٹس کہلاتی ہیں۔
- جیسے ہم main() سے باقی فنکشنز کو کال کر سکتے ہیں ایسے ہی ایک یوزر ڈیفائنڈ فنکشن سے دوسرے یوزر ڈیفائنڈ فنکشن کو بھی کال کر سکتے ہیں۔

## مشق

## سوال نمبر 1: کثیر الانتخابی سوالات۔

- (1) فنکشن بلٹ ان یا \_\_\_\_\_ ہو سکتے ہیں:
- (الف) ایڈمن ڈیفائنڈ (ب) سرور ڈیفائنڈ (ج) یوزر ڈیفائنڈ (د) دونوں الف اور ج
- (2) C- سٹیٹرز ڈلائیری میں موجود فنکشنز \_\_\_\_\_ کہلاتے ہیں:
- (الف) یوزر ڈیفائنڈ (ب) بلٹ ان (ج) ٹکرا پر مبنی (د) تکراری
- (3) فنکشن کو پاس کی گئی قیمتیں \_\_\_\_\_ کہلاتی ہیں:
- (الف) ہاڈیز (ب) ریٹرن ٹائپس (ج) ارے (د) آرگومینٹس
- (4) char cd() {return = 'a';} اس فنکشن میں "char" \_\_\_\_\_ ہے:
- (الف) ہاڈی (ب) ریٹرن ٹائپ (ج) ارے (د) آرگومینٹس
- (5) فنکشنز کو استعمال کرنے کے فوائد \_\_\_\_\_ ہیں:-
- (الف) پڑھے جانے کی صلاحیت (ب) بار بار استعمال (ج) ڈیبگنگ میں آسانی (د) پہلے تینوں
- (6) اگر فنکشن ہاڈی میں تین ریٹرن سٹیٹمنٹس ہوں تو ان میں سے \_\_\_\_\_ چلیں گی:
- (الف) ایک (ب) دو (ج) تین (د) پہلی اور آخری
- (7) پڑھے جانے کی صلاحیت (readability) کوڈ کو \_\_\_\_\_ کرنے میں مدد دیتی ہے:
- (الف) سمجھنے (ب) تبدیل کرنے (ج) ڈیبگ کرنے (د) پہلے تینوں
- (8) \_\_\_\_\_ سے مراد کوڈ ایک اور فنکشن میں ٹرانسفر کرنا ہے:
- (الف) کالنگ (ب) ڈیفائننگ (ج) ری۔ رائٹنگ (د) انکلیوڈنگ (including)

## سوال نمبر 2: درج ذیل کی تعریف کریں۔

- (1) فنکشنز (2) بلٹ۔ ان فنکشنز (3) فنکشن پیرامیٹرز (4) بار بار استعمال (reusability)
- (5) فنکشن کو کال کرنا

## سوال نمبر 3: درج ذیل سوالات کے مختصر جوابات لکھیں۔

- (1) آرگومینٹس اور پیرامیٹرز میں کیا فرق ہے؟ ایک مثال دیں۔
- (2) فنکشن ڈیفینیشن کے حصوں کی فہرست لکھیں۔
- (3) کیا یہ ضروری ہے کہ فنکشن ڈیفینیشن اور فنکشن کال کی ڈیٹا ٹائپس میں ہم آہنگی ہو؟ مثال کے ساتھ جواب کی توثیق کریں۔
- (4) فنکشنز استعمال کرنے کے فوائد کی وضاحت کریں۔
- (5) آپ کی۔ ورڈ return کے بارے میں کیا جانتے ہیں؟

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

- a) `void sum (int a, int b)`  
    {  
        return a + b;  
    }
- b) `void message ();`  
    {  
        printf (“Hope you are fine :)”);  
        return 23;  
    }
- c) `int max (int a; int b)`  
    {  
        if (a > b)  
            return a;  
        return b;  
    }
- d) `int product (int n1, int n2)`  
    return n1\*n2;
- e) `int totalDigits (int x)`  
    {  
        int count = 0;  
        for (int i = x; i >= 1, i = i/10)  
            count++;  
        return count  
    };

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

```

a) int xyz (int n)
    {
        return n + n;
    }
int main()
    {
        int p = xyz(5);
        p = xyz(p);
        printf ("%d ",p);
    }

b) void abc (int a, int b, int c)
    {
        int sum = a + b + c;
    }
int main()
    {
        int x = 4, y = 7, z = 23, sum1 = 0;
        abc (x, y, z);
        printf ("%d %d %d" x, y ,z);
    }

c) int aa (int x)
    {
        int p = x / 10;
        x++;
        p = p + (p * x);
        return p;
    }
int main()
    {
        printf ("We got %d ", aa(aa(23)));
    }

```

```
d) float f3(int n1, int n2)
{
    n1 = n1 + n2;
    n2 = n2 - n1;
    return 0;
}
int main()
{
    printf ("%f\n", f3(3, 2));
    printf ("%f\n", f3(10, 6));
}
```



## پروگرامنگ کی مشقیں

## مشق نمبر 1:

ایک ایٹیجر (integer)  $x$  کا مربع معلوم کرنے کے لیے  $\text{int square}(\text{int } x)$ ; فنکشن لکھیں۔

## مشق نمبر 2:

ایک فنکشن  $\text{int power}(\text{int } x, \text{int } y)$ ; لکھیں جو  $x^y$  معلوم کر کے ریٹرن کرے۔

## مشق نمبر 3:

ایک نمبر کا فیکٹوریل نکلنے کا فنکشن لکھیں۔

## مشق نمبر 4:

ایک فنکشن لکھیں جو مثلث کے تین زاویے لے اور بتائے کہ یہ زاویے صحیح مثلث کے ہیں یا نہیں۔ ایک صحیح مثلث وہ ہوتی ہے جس کے تینوں زاویوں کا مجموعہ 180 ہو۔

## مشق نمبر 5:

ایک فنکشن لکھیں جو رقم اور انٹرسٹ ریٹ لے اور انٹرسٹ کی رقم ریٹرن کرے۔

## مشق نمبر 6:

ایک فنکشن لکھیں جو ایک نمبر ان پٹ لے اور سپیسز کے ساتھ اس کے ہندسے پرنٹ کرے۔

## مشق نمبر 7:

کسی نمبر کا ٹیبل پرنٹ کرنے کا فنکشن لکھیں۔

## اصطلاحات

- :\n یہ بتاتا ہے کہ کرسر کو اگلی لائن کے شروع میں لے کر جانا ہے۔
- :\t یہ بتاتا ہے کہ کرسر کو افقی طور پر اگلے ٹیب سٹاپ پر لے کر جانا ہے۔ ایک ٹیب سٹاپ آٹھ سپیسز کا مجموعہ ہوتا ہے۔
- آرگومینٹس: وہ قیمتیں جو فنکشن کال کے دوران فنکشن کو پاس کی جاتی ہیں۔
- ارتھمیٹک اوپریٹرز: یہ ارتھمیٹک فنکشنز کی قیمت نکالنے کے لیے ڈیٹا پر حساب کتاب کرنے میں استعمال ہوتے ہیں۔ +, -, \*, /, %
- ارتھمیٹک اوپریٹرز ہیں۔
- ارے کی انیشیلائزیشن: پہلی مرتبہ ارے میں قیمتیں لکھنا۔ ایک ارے کو ڈیکلیریشن کے وقت یا اس کے بعد انیشیلائز کیا جاسکتا ہے۔
- ارے کا سائز: زیادہ سے زیادہ اریٹیمینٹس کی تعداد جو ایک ارے میں رکھے جاسکتے ہیں۔
- ارے: ایک ڈیٹا سٹرکچر جو کمپیوٹر کی میموری میں اکٹھی لوکیشنز پر ایک ہی ڈیٹا ٹائپ کی ایک سے زیادہ قیمتیں رکھ سکتا ہے۔
- اسائنمنٹ اوپریٹرز: یہ ایک متغیر میں قیمت رکھنے یا ایک متغیر کو دوسرے متغیر کی قیمت منسوب کرنے کے لیے استعمال ہوتا ہے۔
- ایڈیٹر: ایک سافٹ ویئر جس کے ذریعے پروگرامر کمپیوٹر پروگرام لکھ سکتا ہے اور اس میں ترمیم کر سکتا ہے۔
- اسکیپ سیکوئنس: یہ printf کو بتاتا ہے کہ عام طریقہ کار سے ہٹ کر کام کرنا ہے۔
- یہ اسکیپ کریکٹر (\) اور ایسے کریکٹر کا مجموعہ ہوتا ہے جس سے خاص فنکشن نیٹی منسوب ہو۔
- انڈیکس: یہ ارے کے اریٹیمینٹس تک رسائی حاصل کرنے کے لیے استعمال ہوتا ہے۔ متغیرات کو بھی ارے انڈیکسز کے طور پر استعمال کیا جاسکتا ہے۔
- انٹچر ڈیٹا ٹائپ: ایک ڈیٹا ٹائپ جس میں انٹچر کا اریٹیمینٹس محفوظ کیے جاتے ہیں۔ اس کا سائز 4 بائٹس ہے۔
- انٹیگرل یٹ ڈیویڈیو پلیمینٹ انوائزمنٹ: ایک ایسا سافٹ ویئر جو پروگرامر کو کمپیوٹر پروگرام لکھنے اور چلانے کے لیے پروگرامنگ انوائزمنٹ فراہم کرے۔
- if سٹیٹمنٹ: یہ کنڈیشن سے منسوب کوڈ تہ چلاتی ہے جب کنڈیشن پوری ہو ورنہ نہیں چلاتی۔
- if-else سٹیٹمنٹ: یہ if سٹیٹمنٹ سے منسوب سٹیٹمنٹس کا سیٹ چلاتی ہے۔ اگر کنڈیشن پوری ہو ورنہ else سٹیٹمنٹ سے منسوب سٹیٹمنٹس کا سیٹ چلاتی ہے۔
- بنیادی اوپریٹرز: ان میں ارتھمیٹک اوپریٹرز، اسائنمنٹ اوپریٹرز، ریلیشنل اوپریٹرز اور منطقی اوپریٹرز شامل ہیں۔
- بانسز اوپریٹرز: انھیں دو اوپریٹرز درکار ہوتے ہیں۔
- بولین: ایک ڈیٹا ٹائپ جس میں true یا false رکھا جاسکتا ہے۔
- بلٹ ان فنکشنز: وہ فنکشنز جو C کی سٹینڈرڈ لائبریری میں موجود ہوں۔
- پیرامیٹرز: مختلف ڈیٹا ٹائپس کے متغیرات جن میں فنکشن کو پاس کی گئی قیمتیں رکھی جاتی ہیں۔
- پروگرامر: وہ شخص جسے معلوم ہو کہ ایک صحیح کمپیوٹر پروگرام کیسے لکھا جاتا ہے۔
- پروگرامنگ انوائزمنٹ: پروگرامنگ کے تمام اہم آلات کا مجموعہ۔
- پروگرامنگ لیٹگوئجز: وہ خاص زبانیں جن میں پروگرام لکھتے ہیں۔

## اصطلاحات

ترجیح: اس سے معلوم ہوتا ہے کہ کونسا اوپریشن پہلے انجام دینا ہے۔

ٹرنری اوپریٹرز: انھیں تین اوپریٹرز کا رہتے ہیں۔

ڈیٹا سٹرکچر: ایک کنٹینر جس میں ایک خاص ترتیب سے ڈیٹا آئٹمز کے مجموعے کو محفوظ کیا جاتا ہے۔

ڈیٹا ٹائپ: یہ بتاتی ہے کہ متغیر میں کس قسم کی قیمت محفوظ کی جاسکتی ہے۔

ریلیشنل اوپریٹرز: یہ دو قیمتوں میں موازنہ کر کے ان کا تعلق بتاتے ہیں۔

ریٹرن ٹائپ: یہ اس قیمت کی ڈیٹا ٹائپ ہے جو فنکشن ریٹرن کرتا ہے۔

سیکونڈری کنٹرول: تمام سٹیٹمنٹس دی گئی ترتیب سے چلائی جاتی ہیں۔

سٹیٹمنٹ ٹرمینل: ایک شناخت کنندہ جو کمپائلر کو بتانا ہے کہ سٹیٹمنٹ ختم ہوگئی ہے۔ C- لینگویج میں سیمی کولن (;) بطور سٹیٹمنٹ ٹرمینل استعمال

ہوتا ہے۔

سٹرنگ: کریکٹر کا مجموعہ۔

سنٹیکس: ہر پروگرامنگ لینگویج کے کچھ ابتدائی تعمیراتی عناصر اور پروگرام لکھنے کے کچھ اصول ہوتے ہیں۔ اصولوں کے اس سیٹ کو لینگویج

سنٹیکس (Language Syntax) کہتے ہیں۔

شارٹ سرکٹنگ: پورے ایکسپریشن پر کام کیے بغیر اوپریشن کا جواب نکالنا۔

شناخت کنندہ: ایک متغیر کا حوالہ دینے کے لیے استعمال کیا جانے والا نام۔

فنکشن کی باؤی: یہ ان سٹیٹمنٹس کا سیٹ ہوتا ہے۔ جنہیں چلا کر فنکشن مخصوص کام سرانجام دیتا ہے۔

فنکشن کو کال کرنا: اس فنکشن کو کنٹرول ٹرانسفر کرنا۔

فلوئنگ پوائنٹ: ایک ڈیٹا ٹائپ جو چھ ہندسوں تک پر سائز ریئل کانسٹنٹ (Precise Real Constant) محفوظ کرنے کے

لیے استعمال ہوتی ہے۔ اس کا سائز 4 بائٹس ہے۔

فارمیٹ سپسفاٹرز: یہ ان پٹ آؤٹ پٹ اوپریٹرز کے دوران ڈیٹا ٹائپ کا فارمیٹ بنانے کے لیے استعمال ہوتے ہیں۔

فنکشن سگنچر: فنکشن کی ان پٹ اور آؤٹ پٹ کی وضاحت کرتا ہے۔

فنکشن: سٹیٹمنٹس کا ایک بلاک جو ایک خاص کام سرانجام دیتا ہے۔

Getch() فنکشن: یہ صارف سے ایک کریکٹر لینے کے لیے استعمال ہوتا ہے اسے صرف کریکٹر ان پٹ دی جاسکتی ہے درج کیا گیا کریکٹر

سکرین پر ظاہر نہیں ہوتا۔

کریکٹر: ایک ڈیٹا ٹائپ جس میں صرف کریکٹر محفوظ کیے جاسکتے ہیں۔ اس کا سائز بائٹ ہوتا ہے۔

گمنٹس: وہ سٹیٹمنٹس جو چلتی نہیں ہیں۔

کمپائلر: ایک ایسا سافٹ ویئر جو کسی ہائی لیول کی پروگرامنگ لینگویج میں لکھے گئے کوڈ کو ایسے کوڈ میں تبدیل کر دیتا ہے جسے مشین سمجھ سکے۔

کمپیوٹر پروگرام: ایک خاص کام کو کرنے کے لیے انسان کی لکھی گئی ہدایات کی فہرست۔

## اصطلاحات

- کمپیوٹر پروگرامنگ: کمپیوٹر پروگرام کی ہدایات کو کمپیوٹر میں محفوظ کرنے کا عمل۔
- کنڈیشن: اس میں کوئی بھی درست ایکسپریشن آسکتا ہے جیسے اریٹھمیک ایکسپریشنز، ریلیشنل ایکسپریشنز، منطقی ایکسپریشنز یا ان سب کا مجموعہ۔
- کامپٹینٹس: وہ ڈیٹا جسے مزید پروسیسنگ کے لئے کمپیوٹر کی میموری میں محفوظ کیا جاتا ہے۔
- کنٹرول سٹریٹجی: وہ پروگرام جو تسلسل کو کنٹرول کرتی ہے۔
- کی۔ ورڈز: پروگرامنگ لیگنوج میں پہلے سے ڈیفائن کیے ہوئے الفاظ کی فہرست۔
- ہیڈرفائلز: وہ فائلز جن میں ڈیزائنرز نے موجودہ فنکشنز ڈیفائن کیے ہوں۔
- ہیڈر سیکشن: وہ سیکشن جس میں ہیڈرفائلز شامل کی جاتی ہیں۔
- لوپ سٹرکچر: یہ سٹریٹجی کے ایک سیٹ کو بار بار دہرانے کے لیے استعمال ہوتا ہے۔
- مشروط سٹریٹجی: وہ سٹریٹجی جو شرائط کی بنا پر یہ فیصلہ کرنے میں مدد دیتی ہیں کہ آگے کوئی سٹریٹجی چلائی جائے۔
- منطقی AND اوپریٹر: یہ true جواب دیتا ہے اگر دونوں طرف کے ایکسپریشنز true ہوں۔
- منطقی NOT اوپریٹر: یہ true جواب دیتا ہے اگر ایکسپریشن false ہو اور false جواب دیتا ہے اگر ایکسپریشن true ہو۔
- منطقی اوپریٹر: یہ بولین ایکسپریشنز پر اوپریٹیشن انجام دیتے ہیں اور جواب میں بولین قیمت واپس کرتے ہیں۔
- مین فنکشن: یہاں سے پروگرام چلنا شروع ہوتا ہے۔
- ماڈولس اوپریٹر: ایک بائری اوپریٹرز جو بائیں اوپریٹرز کو دائیں اوپریٹرز پر تقسیم کرتا ہے اور تقسیم کے بعد بچنے والی رقم واپس کرتا ہے۔
- متغیر کی ڈیکلیریشن: متغیر کا نام اور ڈیٹا ٹائپ متعین کرنا۔
- متغیر کی اینیلازیشن: پہلی مرتبہ ایک متغیر سے قیمت منسوب کرنا۔
- متغیرات: وہ کنڈیشنز جن میں کامپٹینٹس یا قیمتیں محفوظ کی جاتی ہیں۔
- پرنٹ سٹریٹجی: مشروط سٹریٹجی میں مشروط سٹریٹجی
- printf() فارمیٹڈ آؤٹ پٹ سکریں پر دکھانے کا بلٹ ان فنکشن۔
- scanf(): صارف سے فارمیٹڈ ان پٹ ریڈ کرنے والا C لیگنوج کا بلٹ ان فنکشن۔
- یوزری اوپریٹرز: انھیں صرف ایک اوپریٹرز درکار ہوتا ہے۔
- یوزر ڈیفائنڈ فنکشنز: وہ فنکشنز جنہیں پروگرامر خود ڈیفائن کرے۔

## انڈیکس

(ک)	(ت)	(الف)
کریکٹر کانسٹیٹ، 34	ترجیح، 41	AND اوپریٹر، 39
کنٹرول سٹریٹجی، 52	(ج)	آرگومینٹس، 108
کمپیوٹر پروگرام، 2	جمع کا اوپریٹر، 35	ارٹھمیٹک اوپریٹرز، 32
کمپیوٹر پروگرام، 2	(ڈ)	ارے کی ڈیکریشن، 79
28.conio.h	ڈیٹا سٹرکچر، 78	ارے کی انیشیلائزیشن، 79
کنسول، 5	ڈیٹا ٹائپ، 116	انڈیکس، 80
کانسٹیٹس، 10	(ذ)	آپٹر کانسٹیٹ، 10
(گ)	ذخیرہ الفاظ، 6	آپٹر، 12
28.getch()	(ر)	انٹی گریٹڈ یوٹیلٹیٹ انوائزمنٹ <sup>3</sup>
(ل)	ریئل کانسٹنٹ، 10	اوپریٹر، 31
لائک سیکشن، 6	ریشل کریکٹرز، 37	ان سائنٹڈ int، 12
لوپ سٹرکچر، 83	ریزن ویلیو، 104	OR اوپریٹر، 40
(م)	(س)	ارے کا سائز، 79
منسوب کردہ کوڈ، 53	سائنٹڈ int، 12	ارے، 78
مین فنکشن کی باڈی، 7	سنگل لائن کمنٹ، 8	اسائنمنٹ اوپریٹر، 316
مطلوبہ الفاظ، 6	سٹیٹمنٹ ٹرمینیٹر، 29	ایڈیٹر، 4
منطقی اوپریٹرز، 39	سٹرنگ، 12	اسکیپ کریکٹر، 29
مین فنکشن، 7	سٹیکس ایرر، 5	ان پٹ، آؤٹ پٹ اوپریٹرز، 23
مین فنکشن، 7	سٹیکس، 5	if سٹیٹمنٹس، 53
ماڈولس آپریٹر، 36	(ش)	if-else سٹیٹمنٹس، 59
	شناخت کنندہ، 11	(ب)
	شرط، 53	بانٹری اوپریٹر، 41
	(ض)	بولین، 12
	ضرب کا آپریٹر، 34	بلٹ ان فنکشن، 103
	(ف)	بار بار استعمال کی صلاحیت، 104
	فلوئنگ پوائنٹ، 12	(پ)
	فارمیٹ سپیفاؤر، 24	پیرامیٹرز، 104
	فنکشن کال، 107	printf فنکشن، 23
	فنکشن ڈیفینیشن، 105	پروگرامر، 2
	فنکشن سگنچر، 105	پروگرامنگ انوائزمنٹ، 2
	فنکشن، 103	پروگرامنگ لیگلوٹیج، 2

## جوابات

## باب 1:

سوال نمبر 1: کثیر الانتخابی سوالات۔

- |       |        |
|-------|--------|
| ج -6  | ج -1   |
| ب -7  | الف -2 |
| ب -8  | ب -3   |
| ب -9  | ب -4   |
| ج -10 | الف -5 |

سوال نمبر 2: درست/غلط۔

- 1- درست
- 2- درست
- 3- غلط
- 4- غلط
- 5- درست

سوال نمبر 3: کالم ملائیں۔

- |   |    |
|---|----|
| d | -1 |
| f | -2 |
| a | -3 |
| e | -4 |
| g | -5 |
| b | -6 |
| h | -7 |
| e | -8 |

## جوابات

## باب:2

سوال نمبر 1: کثیر الانتخابی سوالات۔

- |       |      |
|-------|------|
| د -6  | د -1 |
| ب -7  | ج -2 |
| ب -8  | ج -3 |
| ج -9  | ب -4 |
| د -10 | ب -5 |

سوال نمبر 2: درست/غلط۔

- |        |         |
|--------|---------|
| 3- غلط | 1- غلط  |
| 4- غلط | 2- درست |
|        | 5- غلط  |

سوال نمبر 3: آؤٹ پٹ۔

- |       |    |
|-------|----|
| 0 7 9 | -1 |
| nn    | -2 |

nnn

n

t nn /n/n nn/n

9 -3

5 -4

1 -5

## باب:3

سوال نمبر 1: کثیر الانتخابی سوالات۔

- 1- الف -2 د -3 ج -4 الف -5 ب -6 د -7 الف -8 ج

سوال نمبر 5: آؤٹ پٹ۔

1.  $a = 17, b = 10$
2. Hope for the Best
3.  $6 < 9$  and  $N = N$
4.  $a=50176, b=224, c=22, d=484$
5.  $x = 16$   
 $x = 16, y = 8, z = 9$

## جوابات

## باب 4:

سوال نمبر 1: کثیر الانتخابی سوالات۔

- 6- د
- 7- د
- 8- ج
- 9- الف
- 10- الف

- 1- ج
- 2- الف
- 3- الف
- 4- الف
- 5- ب

سوال نمبر 5: آؤٹ پٹ۔

1. Sum is 25
2. \*
3. z = 50  
z = 49  
z = 48  
i = 50
5. 0.000000  
0.000000  
0.000000  
2.640000  
5.520000  
11.959999

4. 4  
16  
36  
64

## باب 5:

سوال نمبر 1: کثیر الانتخابی سوالات۔

- 5- د
- 6- الف
- 7- د
- 8- الف

- 1- ج
- 2- ب
- 3- د
- 4- ب

سوال نمبر 5: آؤٹ پٹ۔

- 20 -1
- 4 7 23 -2
- We got 260 -3
- 0.000000 -4
- 0.000000