# کمپیوٹر سائنس

# 10

ABL
JavaScript
Delphi
Magik
Ada
R
Oz
Java
Harbour
Clipper
C++
D
Python
Objective-C
Clarion
Gambas
Visual Fox
VB.NET
Action Script
C#
Ruby
Visual Objects
PHP5
Class
Perl
VisualBasic 6
Clipper
AJAX

پنجاب کریکولم اینڈ ٹیکسٹ بک بورڈ، لاہور

جملہ حقوق بحق پنجاب کریکولم اینڈ ٹیکسٹ بُک بورڈ، لاہور محفوظ ہیں۔
اس کتاب کا کوئی حصہ نقل یا ترجمہ نہیں کیا جا سکتا اور نہ ہی اسے ٹیسٹ پیپر، گائیڈ بُکس، خلاصہ جات،
نوٹس یا امدادی کتب کی تیاری میں استعمال کیا جا سکتا ہے۔

# فہرست

| باب نمبر | عنوانات | صفحہ نمبر |
|---|---|---|
| 1 | پروگرامنگ کا تعارف | 2 |
| 2 | یوزر انٹریکشن (User Interaction) | 21 |
| 3 | مشروط منطق (Conditional Logic) | 51 |
| 4 | ڈیٹا اینڈ ریپیٹیشن (Data and Repetition) | 77 |
| 5 | فنکشنز (Functions) | 102 |
| الف | اِصطلاحات | 118 |
| ب | انڈیکس (اشاریہ) | 121 |
| ج | جوابات | 122 |

**مصنّف:**
ڈاکٹر عدنان ہاشمی
اسسٹنٹ پروفیسر
شعبہ کمپیوٹر سائنس اینڈ آئی۔ٹی
دی یونیورسٹی آف لاہور

**نگران:**
جہانزیب خان
ایس۔ایس (کمپیوٹر سائنس)
پی۔سی۔ٹی۔بی لاہور

**ڈائریکٹر مسوّدات:**
عطا دستگیر

**نظرِ ثانی:**
ڈاکٹر جمیل الرحمٰن
ایس۔ایس۔ایس (اُردو)
پی۔سی۔ٹی۔بی لاہور

**مدیر:**
ڈاکٹر مدثر نصیر
ایسوسی ایٹ پروفیسر
شعبہ کمپیوٹر سائنس اینڈ آئی۔ٹی
دی یونیورسٹی آف لاہور

**ڈپٹی ڈائریکٹر (گرافکس) / آرٹسٹ:**
مسز عائشہ وحید

**لے آؤٹ / ڈیزائننگ:**
علیم الرحمٰن

**کمپوزنگ:**
عرفان شاہد
عمیر طارق
محمد ندیم

| تاریخ اشاعت | ایڈیشن | طباعت | تعداد اشاعت | قیمت |
|---|---|---|---|---|
| دسمبر 2020ء | تجرباتی | اول | | |

# فہرست

یونٹ 1: پروگرامنگ کا تعارف 1

1.1 پروگرامنگ انوائرنمنٹ 2

- 1.1.1 انٹیگریٹڈ ڈیویلپمنٹ انوائرنمنٹ..(Integrated Development Environment) ..... 3
- 1.1.2 ٹیکسٹ ایڈیٹر..(Text Editor) ..... 4
- 1.1.3 کمپائلر. (Compiler) ..... 5

1.2 پروگرامنگ کے بنیادی نِکات 5

- 1.2.1 مختص کیے گئے الفاظ ..... 6
- 1.2.2 C-پروگرام کی ساخت ..... 6
- 1.2.3 C-پروگرام میں کمنٹس کا سنٹیکس اور مقصد ..... 8

1.3 مستقلات اور متغیّرات 9

- 1.3.1 مستقلات ..... 10
- 1.3.2 متغیّرات ..... 11
- 1.3.3 متغیّر کی ڈیٹا ٹائپ ..... 11
- 1.3.4 متغیّرات کا نام ..... 12
- 1.3.5 متغیّر کی ڈکلیریشن. (Variable Declaration) ..... 13
- 1.3.6 متغیّر کی انشلائزیشن.(Variable Initialization) ..... 14

یونٹ 2: یوزر انٹریکشن 21

2.1 اِن پٹ/آؤٹ پٹ (I/O) فنکشن 23

- 2.1.1 printf( ) ..... 23
- 2.1.2 فارمیٹ سپیسفائر..(Formate Specifier) ..... 24
- 2.1.3 scanf( ) ..... 26
- 2.1.4 getch( ) ..... 28
- 2.1.5 سٹیٹمنٹ ٹرمینیٹر (Statement Terminator) ..... 29
- 2.1.6 اسکیپ سیکوئنس.(Escape Sequence) ..... 29

2.2 اوپریٹرز 31

- 2.2.1 اسائنمنٹ اوپریٹر..(Assignment Operators) ..... 31
- 2.2.2 ارتھمیٹک اوپریٹر.....(Arithmatic Operators) ..... 32
- 2.2.3 ریلیشنل اوپریٹر.(Relational Operators) ..... 37
- 2.2.4 اسائنمنٹ اوپریٹر (=) اور برابر کا اوپریٹر (==) ..... 38
- 2.2.5 منطقی اوپریٹر ..... 39
- 2.2.6 یونری بمقابلہ بائنری اوپریٹرز...(Uniry vs Binary Operators) ..... 41

# فہرست

- 2.2.7 اوپریٹرز کی ترجیح ..... 42

**یونٹ 3: مشروط منطق** 51

3.1 کنٹرول سٹیٹمنٹس (Control Statements) 52

3.2 سلیکشن سٹیٹمنٹس (Selection Statement) 53

- 3.2.1 if سٹیٹمنٹ ..... 59
- 3.2.2 if-else سٹیٹمنٹ ..... 64
- 3.2.3 نیسٹڈ سلیکشن سٹرکچرز (Nested Selection Structures) ..... 64
- 3.2.4 حل شُدہ مثالیں ..... 77

**یونٹ 4: ڈیٹا اینڈ ریپیٹیشن** 78

4.1 ڈیٹا سٹرکچرز 78

- 4.1.1 ارے (Array) ..... 78
- 4.1.2 ارے ڈکلیریشن (Array Declaration) ..... 79
- 4.1.3 ارے انشلائزیشن (Array Initialization) ..... 79
- 4.1.4 ارے ایلمنٹس تک رسائی ..... 80
- 4.1.5 ویری ایبلز کا بطور ارے انڈیکس استعمال ..... 82

4.2 لوپ سٹرکچر 83

- 4.2.1 لوپس کا عام سٹرکچر ..... 83
- 4.2.2 لوپس کا عام سنٹیکس ..... 83
- 4.2.3 نیسٹڈ لوپس (Nested Loops) ..... 87
- 4.2.4 حل شُدہ مثالیں ..... 91
- 4.2.5 لوپس اور اریز (Loops and Arays) ..... 93
- 4.2.6 حل شُدہ مثالیں ..... 95

**یونٹ 5: فنکشنز** 101

5.1 فنکشنز 102

- 5.1.1 فنکشنز کی اقسام ..... 102
- 5.1.2 فنکشنز کے فوائد ..... 103
- 5.1.3 فنکشن کا سیگنیچر (Signature of Function) ..... 103
- 5.1.4 فنکشن کو ڈیفائن کرنا ..... 104

**6۔ اصطلاحات** 118

**7۔ انڈیکس (اشاریہ)** 121

**8۔ جوابات** 122

## باب 1

# پروگرامنگ کا تعارف

## تدریسی مقاصد:(Students Learning Outcomes)

**انٹیگریٹڈ ڈیویلپمنٹ انوائرنمنٹ (IDE) کے تصور کی وضاحت**

- C- پروگرامنگ انوائرنمنٹ کے ان موڈیولز کی وضاحت:
    - ٹیکسٹ ایڈیٹر (Text Editor)
    - کمپائلر (Compiler)
- کی ورڈ (Keyword) کی شناخت کرنا
- C- پروگرام کی ساخت کی وضاحت جس میں یہ شامل ہیں:
    - انکلیوڈ (include)
    - main() فنکشن
    - main() کی باڈی
- کمینٹس (Comments) کا مقصد اور ان کے سنٹیکس کی وضاحت
- مستقل (Constant) اور متغیر (Variable) میں فرق کی وضاحت
- متغیر (Variable) کے نام رکھنے کے اصولوں کی وضاحت
- C - کی درج ذیل ڈیٹا ٹائپس (data types) اور ان میں موجود بائٹس کی تعداد کا علم ہونا
    - انٹیجر (int(signed)/unsigned)
    - فلوٹنگ پوائنٹ (float)
    - کریکٹر (char)
- متغیرات (Variables) کو واضح (Declare) اور اجرا کرنے (initialize) کے پروسیس کی وضاحت

## تعارف (Introduction)

کمپیوٹر ہماری زندگی کا ایک اہم حصہ بن چکے ہیں۔ یہ ہمیں ریاضی کے پیچیدہ سوالات اور انٹرنیٹ پر کھوج لگانے (Searching) سے لے کر سیٹلائٹ اور راکٹ لانچر کو چلانے اور کنٹرول کرنے تک کئی مسائل کے حل میں مدد دیتا ہے۔ دراصل کمپیوٹر خود سوچنے کی صلاحیت نہیں رکھتا۔ تمام امور سرانجام دینے کے لیے انسان کمپیوٹر کو ہدایت دیتے ہیں جن سے اُنھیں معلوم ہوتا ہے کہ ایک خاص قسم کے مسئلے کو کس طرح حل کرنا ہے۔ ہدایت کی اس سیریز یا فہرست کو کمپیوٹر پروگرام یا سافٹ ویئر (Software) کہتے ہیں اور یہ ہدایات کمپیوٹر میں محفوظ کرنے کے عمل کو کمپیوٹر پروگرامنگ کہتے ہیں۔ وہ شخص جو جانتا ہو کہ ایک کمپیوٹر پروگرام کس طرح لکھا جاتا ہے 'پروگرامر' کہلاتا ہے۔

کمپیوٹر انگریزی، اُردو یا دیگر عام زبانیں نہیں سمجھتے جن کے ذریعے سے انسان ایک دوسرے سے بات چیت کرتے ہیں۔ ان کی اپنی خاص زبانیں ہوتی ہیں جنھیں کمپیوٹر کے سائنس دان (Computer Scientists) ڈیزائن کرتے ہیں۔ پروگرامر ان خاص زبانوں میں کمپیوٹر پروگرام لکھتے ہیں جنھیں پروگرامنگ لینگوئجز (Programming Languages) کہا جاتا ہے۔ بہت زیادہ استعمال ہونے والی پروگرامنگ لینگوئجز میں سے چند ایک C#, C, C++, JAVA اور Python ہیں۔ اس کتاب میں ہم کمپیوٹر پروگرام لکھنے کے لیے C- لینگوئج استعمال کریں گے۔ اس باب میں C- لینگوئج کو استعمال کرتے ہوئے کمپیوٹر پروگرامنگ کی کچھ بنیادی چیزوں کا ذکر کیا گیا ہے۔

**کیا آپ جانتے تھے؟**

C- لینگوئج 1949ء اور 1973ء کے درمیان بیل لیبز (Bell Labs) میں ڈینس رِچی (Dennis Ritchie) نے تیار کی تھی۔

## 1.1 پروگرامنگ انوائرنمنٹ (Programming Environment)

کسی بھی کام کو سرانجام دینے کے لیے ہمیں مناسب ٹولز (Tools) کی ضرورت ہوتی ہے۔ مثلاً باغبانی کرنے کے لیے باغبانی کے آلات اور پینٹنگ کے لیے پینٹ، برش اور کینوس (Canvas) کی ضرورت ہوتی ہے۔ اسی طرح پروگرامنگ کے لیے بھی خاص ٹولز درکار ہوتے ہیں۔ پروگرامنگ کے تمام اہم آلات کو اکٹھا کرنے سے پروگرامنگ انوائرنمنٹ بنتی ہے۔ پروگرام لکھنے سے پہلے پروگرامنگ انوائرنمنٹ تیار کرنا ضروری ہے۔ پروگرام لکھنے اور چلانے کے لیے یہ ہمیں بنیادی پلیٹ فارم فراہم کرتی ہے۔

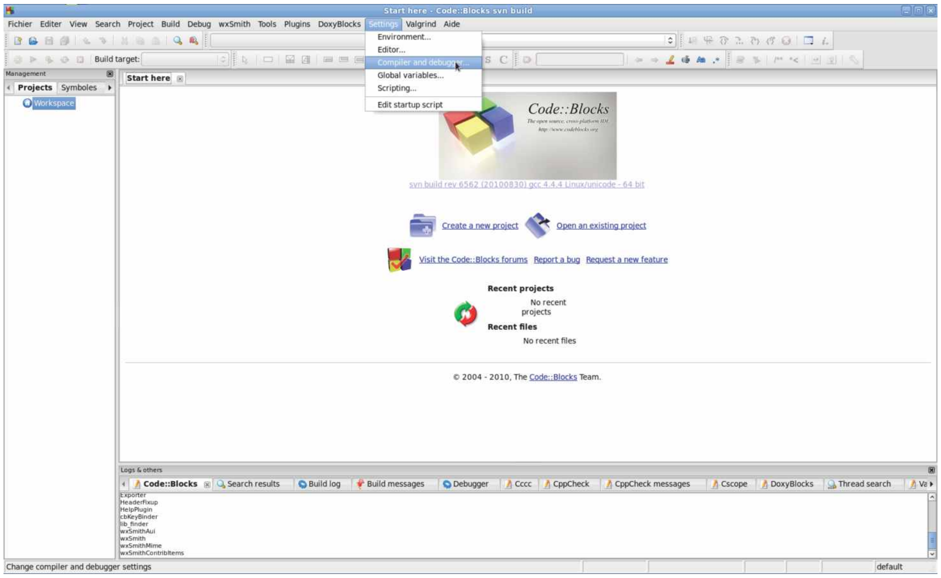
## 1.1.1 انٹیگریٹڈ ڈویلپمنٹ انوائرنمنٹ IDE (Integrated Development Environment)

ایک سافٹ ویئر جو پروگرامر کو کمپیوٹر پروگرام لکھنے اور چلانے میں مدد دینے کے لیے پروگرامنگ انوائرنمنٹ فراہم کرتا ہے وہ **انٹیگریٹڈ ڈویلپمنٹ انوائرنمنٹ (IDE)** کہلاتا ہے۔

IDE کا ایک گرافکل یوزر انٹرفیس (Graphical User Interface) ہوتا ہے جس کی ونڈوز (windows) اور بٹن استعمال کر کے صارف (user) ان پٹ (input) دے سکتا ہے اور آؤٹ پٹ لے سکتا ہے۔ IDE میں ایسے آلات ہوتے ہیں جو پروگرامر کو کمپیوٹر پروگرام لکھنے، چلانے اور ٹیسٹ کرنے کے مراحل میں مدد دیتے ہیں۔ ایک ہی انٹرفیس میں ٹیکسٹ ایڈیٹرز (Text Editors)، کمپائلرز اور ڈیبگرز (Debuggers) کو اکٹھا کر کے یہ مقاصد حاصل کیے جاتے ہیں۔ C- پروگرامنگ لینگوئج کے لیے موجود کچھ IDEs یہ ہیں:

1) Visual Studio
2) X Code
3) Code:: Blocks
4) Dev C++

شکل **1.1** میں Code:: Blocks کی main سکرین دکھائی گئی ہے۔


Start here - Code::Blocks svn build
Fichier Editer View Search Project Build Debug wxSmith Tools Plugins DoxyBlocks Settings Valgrind Aide
Environment...
Editor...
Compiler and debugger...
Global variables...
Scripting...
Edit startup script
Build target:
Management
Projects Symboles
Workspace
Start here
Code::Blocks
svn build rev 6562 (20100830) gcc 4.4.4 Linux/unicode - 64 bit
Create a new project
Open an existing project
Visit the Code::Blocks forums
Report a bug
Request a new feature
Recent projects
No recent projects
Recent files
No recent files
© 2004 - 2010, The Code::Blocks Team.
Logs & others
Code::Blocks
Search results
Build log
Build messages
Debugger
Cccc
CppCheck
CppCheck messages
Cscope
DoxyBlocks
Thread search
Change compiler and debugger settings
default

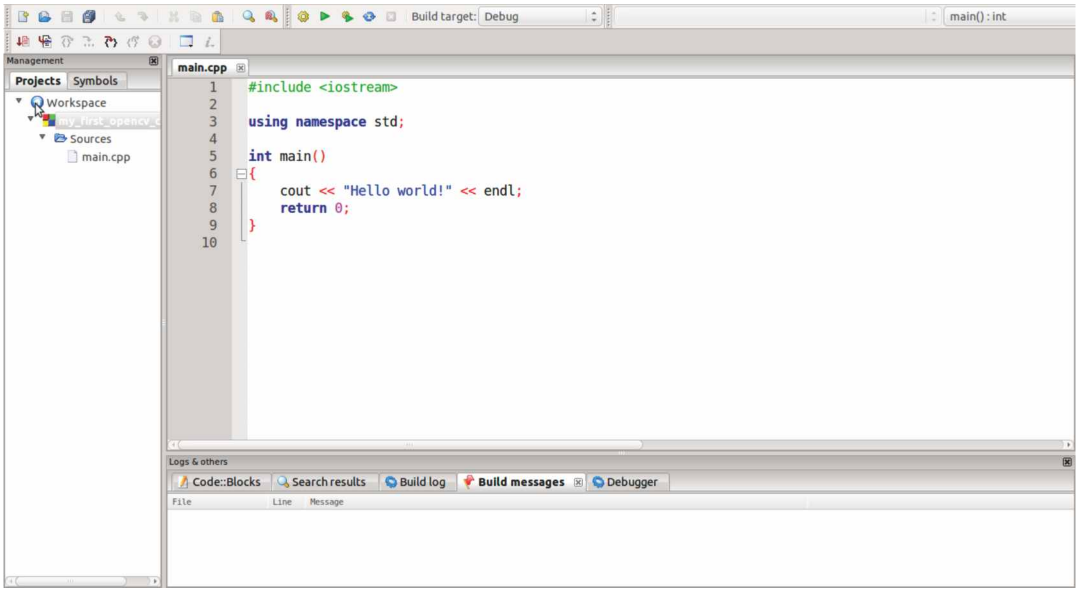
شکل 1.1 Code:: Blocks کا main انٹرفیس۔

**سرگرمی 1.1:**

اپنے ویب براؤزر (Web Browser) کے ذریعے C- پروگرامنگ لینگوئج کی تین مختلف IDEs کے نام معلوم کریں۔

## 1.1.2 ٹیکسٹ ایڈیٹر (Text Editor)

ٹیکسٹ ایڈیٹر ایسا سافٹ ویئر ہے جو پروگرامر کو کمپیوٹر پروگرام لکھنے اور اس میں ترمیم کرنے میں مدد دیتا ہے۔ تمام IDEs کے اپنے ٹیکسٹ ایڈیٹرز ہوتے ہیں۔ ہم IDE کی مین سکرین پر اپنے پروگرام لکھ سکتے ہیں۔

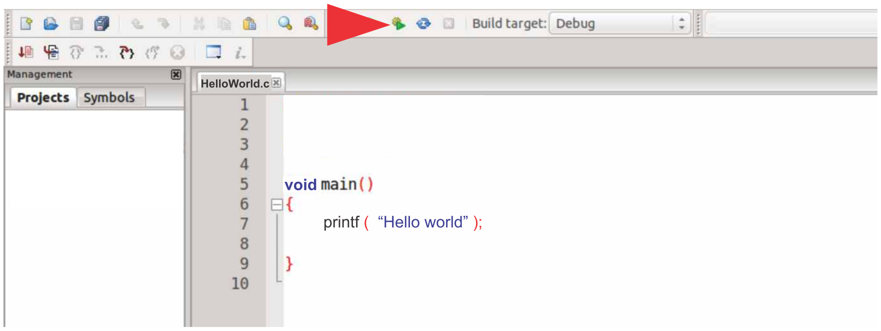


```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

Build target: Debug
main() : int
Management
Projects
Symbols
Workspace
my_first_opencv_
Sources
main.cpp
Logs & others
Code::Blocks
Search results
Build log
Build messages
Debugger
File
Line
Message

شکل 1.2 Code:: Blocks کا ٹیکسٹ ایڈیٹر

شکل 1.2 میں Code:: Blocks IDE کے ٹیکسٹ ایڈیٹر میں لکھا ہوا C- لینگوئج کا ایک بنیادی پروگرام دکھایا گیا ہے۔ جب یہ پروگرام ایگزیکیوٹ ہوتا ہے تو کمپیوٹر سکرین پر "Hello World" ظاہر ہوتا ہے۔ اسے ایگزیکیوٹ کرنے سے پہلے فائل کو محفوظ کرنا ضروری ہے۔ ہم نے اپنے پروگرام کی فائل کا نام "Hello World.c" رکھا ہے۔ پروگرام کی آؤٹ پٹ دیکھنے کے لیے ہم "build and run" بٹن کو کلک کر سکتے ہیں جس کی شکل 1.3 میں نشان دہی کی گئی ہے۔

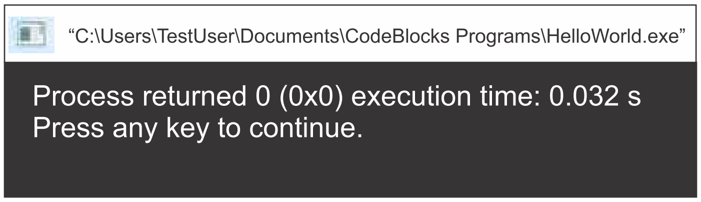


```
void main()
{
        printf (  “Hello world” );
}
```

Build target: Debug
Management
Projects
Symbols
HelloWorld.c

شکل 1.3 Code:: Blocks میں ایگزیکیوٹ ہوتا ہوا پروگرام

کنسول آؤٹ پٹ سکرین (Console output screen) اس طرح نظر آتی ہے جیسے **شکل 1.4** میں دکھائی گئی ہے۔



```
"C:\Users\TestUser\Documents\CodeBlocks Programs\HelloWorld.exe"
Process returned 0 (0x0) execution time: 0.032 s
Press any key to continue.
```

شکل 1.4 پروگرام آؤٹ پٹ

**سرگرمی 1.2:**

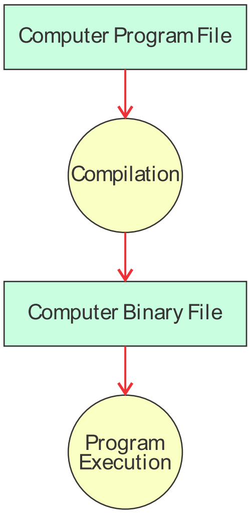
اپنی لیب کے کمپیوٹر میں انسٹال کی ہوئی IDE کھولیں۔ شکل 1.2 میں دیا ہوا پروگرام اپنی IDE کے ٹیکسٹ ایڈیٹر میں لکھیں اور اسے ایگزیکیوٹ کریں۔

### 1.1.3 کمپائلر (Compiler)

کمپیوٹر صرف مشین کی زبان جو کہ 0 اور 1 پر مبنی ہے کو سمجھتا اور اِس میں کام کرتا ہے۔

ایک پروگرام کو ایگزیکیوٹ کرنے کے لیے اِسے مشین کی زبان میں تبدیل کرنا ضروری ہوتا ہے۔

کمپائلر ایسا پروگرام ہے جو کہ ایک اعلیٰ درجے کی زبان میں لکھے گئے پروگرام کو مشین کی زبان یا کوڈ میں تبدیل کرنے کا ذمہ دار ہوتا ہے۔



شکل 1.5: پروگرام ایگزیکیوشن

## 1.2 پروگرامنگ کے بنیادی نِکات (Programming Basics)

ہر پروگرامنگ لینگوئج کے کچھ ابتدائی تعمیراتی عناصر ہوتے ہیں اور اس میں درست پروگرام لکھنے کے لیے کچھ اصول دیے گئے ہوتے ہیں۔ اصولوں کے اس مجموعے کو لینگوئج کا سینٹیکس (Syntax) کہتے ہیں۔ سینٹیکس "پروگرامنگ لینگوئج" کی گرامر کی طرح ہے۔ پروگرامنگ کرتے ہوئے اگر پروگرامنگ لینگوئج کا سینٹیکس یا اصول درست طرح استعمال نہ کیے جائیں تو پروگرام کمپائل (Compile) نہیں ہوتا۔ اس صورت میں ایک ایرر (error) آتا ہے جسے سینٹیکس ایرر کہا جاتا ہے۔

## 1.2.1 مختص کیے گئے الفاظ (Reserved words)

ہر پروگرامنگ لینگوئج میں پہلے سے واضح الفاظ کی ایک فہرست ہوتی ہے۔ ہر لفظ کا ایک خاص مطلب ہوتا ہے جو کمپائلر کو پہلے سے معلوم ہوتا ہے۔ ان الفاظ کو مخصوص کیے گئے الفاظ **(Reserved words)** یا کی-ورڈز **(Keywords)** کہا جاتا ہے۔ اگر پروگرامر ان کی خود کوئی تعریف (Definition) دے تو سینٹیکس ایرر آتا ہے۔

| | | | |
|---|---|---|---|
| auto | double | int | struct |
| break | else | long | switch |
| case | enum | register | typedef |
| char | extern | return | union |
| const | float | short | unsigned |
| continue | for | signed | void |
| default | goto | sizeof | volatile |
| do | if | static | while |

ٹیبل 1.1 میں پروگرامنگ لینگوئج کے ریزرو ورڈز کی فہرست دی گئی ہے۔

**سرگرمی 1.3:**

ان الفاظ میں سے C- لینگوئج کے ریزرو ورڈز یا مختص کیے گئے الفاظ کے گرد دائرہ لگائیں۔

**int, pack, create, case, return, small, math, struct, program, library.**

## 1.2.2 C- پروگرام کی ساخت (Structure of Program)

ہم شکل 1.2 میں لکھے ہوئے پروگرام کو دیکھ کر C- پروگرام کی ساخت کو سمجھ سکتے ہیں۔ ہم دیکھ سکتے ہیں کہ ایک پروگرام کو تین حصوں میں تقسیم کیا جا سکتا ہے۔

### 1- لنک سیکشن یا ہیڈر سیکشن (Link section or Header Section)

C- لینگوئج میں پروگرام لکھتے ہوئے ہم لینگوئج میں پہلے سے ڈیفائن کیے گئے فنکشنز کا بہت زیادہ استعمال کرتے ہیں لیکن ان فنکشنز کو استعمال کرنے کے لیے ہمیں ان فائلوں کو شامل کرنا پڑتا ہے جن میں ان فنکشنز کو بیان کیا گیا ہے۔ یہ فائلیں ہیڈر فائلز (Header files) کہلاتی ہیں۔ ہم پروگرام کے شروع میں include statements لکھ کر اپنے پروگرام میں یہ ہیڈر فائلز شامل کرتے ہیں۔

include statement سٹیٹمنٹ کی ساخت درج ذیل ہے:

**#include <header-file-name>**

یہاں header-file-name کسی بھی ہیڈر فائل کا نام ہو سکتا ہے۔

درج بالا مثال (**شکل 1.2**) میں ہم نے فائل <stdio.h> شامل کی ہے جس میں ان پٹ اور آؤٹ پٹ فنکشنز سے متعلق معلومات ہیں۔ دیگر بہت سی ہیڈر فائلز ہوتی ہیں مثلاً فائل <math.h> میں ریاضی کے سارے پہلے سے طے شدہ (Pre-defined) فنکشنز موجود ہیں۔

### 2- main سیکشن:

اس میں main() فنکشن ہوتا ہے۔ ہر C پروگرام میں ایک main() فنکشن ضرور ہوتا ہے اور یہیں سے پروگرام چلنا شروع ہوتا ہے۔

### 3-main() فنکشن کی باڈی:

main() کی باڈی ٹیڑھی بریکٹوں (curly braces) {} کے اندر ہوتی ہے۔ تمام سٹیٹمنٹس جو ٹیڑھی بریکٹوں {} کے اندر آتی ہیں وہ main فنکشن کی باڈی میں شامل ہوتی ہیں۔ اوپر دیے گئے پروگرام میں سٹیٹمنٹ printf("Hello World!"); کمپیوٹر سکرین پر "Hello World" پرنٹ کرنے کے لیے پری ڈیفائنڈ فنکشن Printf کا استعمال کرتی ہے۔ ہم اپنے پروگرام میں اور فنکشنز بنا کر انھیں بھی main() فنکشن میں استعمال کر سکتے ہیں۔

### اہم نوٹ:

C- لینگوئج کے سینٹیکس کے اعتبار سے درست پروگرام لکھنے کے لئے درج ذیل نکات کو ذہن میں رکھنا ضروری ہے۔

- C- لینگوئج پروگرام میں سٹیٹمنٹس کی ترتیب وہی ہونی چاہیے جس ترتیب سے ہم عبارات (Statements) کو چلانا چاہتے ہیں۔
- C- لینگوئج کیس سینسٹو (case sensitive) ہے۔ یعنی اگر کوئی ریزرو ورڈ چھوٹے حروف میں ڈیفائن کیا گیا ہے تو اسے بڑے حروف میں نہیں لکھا جا سکتا مثلاً int اور Int مختلف ہیں۔ پہلا ریزرو ورڈ ہے جب کہ دوسرا نہیں ہے۔
- ہر سٹیٹمنٹ کے آخر پر سیمی کولن (;) آتا ہے۔

**سرگرمی 1.4:**

C- پروگرام کے مختلف حصوں کی شناخت کریں۔

```
#include<stdio.h>
#include <conio.h>
void main()
{
        printf(“I am a student of class 10”);
        getch();
}
```

## 1.2.3 C- پروگرام میں کمینٹس کا سینٹیکس اور مقصد

کمینٹس پروگرام کی وہ سٹیٹمنٹس ہوتی ہیں جنھیں کمپائلر نظرانداز کر دیتا ہے اور یہ ایگزیکیوٹ نہیں ہوتیں۔ کمینٹس اپنے کوڈ کی وضاحت کرنے کے لیے ہوتے ہیں اور عموماً فطری زبان جیسا کہ انگریزی میں لکھے جاتے ہیں۔

### کمینٹس لکھنے کا مقصد:

کمینٹس کو پروگرام کی دستاویز کی طرح سمجھا جا سکتا ہے۔ ان کے دو مقاصد ہوتے ہیں:

1- یہ دوسرے پروگرامرز کے کوڈ سمجھنے میں مدد دیتے ہیں۔

2- ان کے ذریعے سے ہم اپنے کوڈ کو لکھنے کے کئی سال بعد بھی آسانی سے سمجھ سکتے ہیں۔

ہمیں ان سٹیٹمنٹس کو ایگزیکیوٹ کرنے کی ضرورت نہیں ہوتی اگر ہم ان سٹیٹمنٹس کو ایگزیکیوٹ کریں تو سینٹیکس ایررز آ سکتے ہیں کیوں کہ یہ سٹیٹمنٹس فطری زبان میں لکھی جاتی ہیں۔

### کمنٹس لکھنے کا سینٹیکس:

C- پروگرامنگ لینگوئج میں کمینٹس کی دو قسمیں ہوتی ہیں:

1- سنگل لائن کمینٹس (Single Line Comments)

2- ملٹی لائن کمینٹس (Multi Line Comments)

سنگل لائن کمینٹس // سے شروع ہوتے ہیں۔ اس لائن پر // کے بعد کچھ بھی لکھا جائے وہ کمینٹ ہی سمجھا جاتا ہے۔ مثلاً // This is a comment

**ملٹی لائن کمینٹس** */ سے شروع ہوتے ہیں اور /* پر ختم ہوتے ہیں۔

/* and */ کے درمیان جو بھی لکھا جائے وہ کمینٹ ہی سمجھا جاتا ہے۔ چاہے ایک سے زیادہ سطروں پر ہو۔ مثلاً

```
/*this is
a multi-line
comment*/
```

درج ذیل کوڈ میں کمینٹس کا استعمال دکھایا گیا ہے۔

**EXAMPLE CODE 1.1**

```
#include <stdio.h>
/*this program displays “I am a student of class 10” on the
output screen*/
void main()
{ //body of main function starts from here
      printf(“I am a student of class 10”);
} //body of main function ends here
```

**سرگرمی 1.5:**

درج ذیل میں سے درست کمینٹس پر (✓) کا نشان لگائیں۔

- *comment goes here*
- /comment goes here/
- %comment goes here%
- /* comment goes
  here*/
- /*comment goes here/
- //comment goes here */

## 1.3 مستقلات اور متغیّرات:

ہر زبان کے حروف کا ایک بنیادی سیٹ (کریکٹر سیٹ) ہوتا ہے جنھیں قابل اطلاق طریقے سے جوڑ کر الفاظ بنائے جاتے ہیں اور ان الفاظ کو جملے بنانے میں استعمال کیا جاتا ہے۔ اسی طرح C- پروگرامنگ لینگویج میں بھی کریکٹر سیٹ ہوتا ہے جس میں یہ سب شامل ہوتے ہیں:

**1)** حروف (A,B,...,Y,Z), (a,b,....y,z)

**2)** ہندسے (0-9)

**3)** خصوصی علامات (~`!@#%^&*()_-+=\{}[]:;"'<>,.?/)

ان حروف، ہندسوں اور خصوصی علامات کو جب قابلِ اطلاق طریقے سے جوڑا جائے تو مستقلات، متغیرات اور کی-ورڈز (جنھیں ریزرو ورڈز بھی کہتے ہیں) بنتے ہیں۔ ہم پہلے ہی کی-ورڈ کے بارے میں پڑھ چکے ہیں۔ اب ہم مستقلات اور متغیرات کے بارے میں پڑھیں گے۔

### 1.3.1 مستقلات (Constants):

مستقلات وہ قیمتیں ہوتی ہیں جنھیں پروگرام تبدیل نہیں کرسکتا۔ مثلاً 5, 75.7, 1500 وغیرہ۔ C-پروگرامنگ لینگوئج میں ابتدائی طور پر مستقلات کی تین قسمیں ہیں:

#### 1) انٹیجر مستقلات (Integer constants):

ان کی قیمتوں میں اعشاریہ نہیں آتا مثلاً 7, 1256, 30100, 55555, -54, 2349 وغیرہ۔ یہ مثبت بھی ہوسکتی ہیں اور منفی بھی۔ اگر قیمت سے پہلے کوئی علامت نہ ہو تو قیمت کو مثبت ہی سمجھا جاتا ہے۔

#### 2) حقیقی مستقلات (Real constants):

ان کی قیمتوں میں اعشاریہ آتا ہے مثلاً: 3.14, 15.3333, 75.0, -1575.76, 7941.2345 وغیرہ۔ یہ بھی مثبت اور منفی ہوسکتی ہیں۔

#### 3) کریکٹر مستقلات (Character Constants):

' ' کے اندر لکھا گیا کوئی بھی چھوٹے یا بڑے حروف، پنکچوایشن (Punctuation) کی علامت یا سپیشل ورڈ مستقل کریکٹر ہوتی ہے مثلاً '5', '7', 'a', 'x', '!', ';' وغیرہ۔

**اہم نکتہ:**

مستقل کریکٹر کے طور پر لکھا گیا ہندسہ جیسے '9'، مستقل انٹیجر کریکٹر کے طور پر لکھے گئے ہندسے جیسے 9 سے مختلف ہوتا ہے۔ ہم دو انٹیجر مستقل کو جمع کرکے ریاضی کے اعتبار سے صحیح جواب حاصل کرسکتے مثلاً $9+8=17$ لیکن کریکٹر مستقلات کو جمع کرکے نہیں مثلاً

$'9'+'8'\neq 17$

**سرگرمی 1.6:**

مندرجہ ذیل مستقلات کی اقسام کی شناخت کریں۔

| | | | | |
|---|---|---|---|---|
| 32.768 | -21 | '*' | 1.2 | 12 |
| '\' | 40.0 | 41 | -12.3 | 'a' |

### 1.3.2 متغیّرات (Variables)

متغیر دراصل میموری لوکیشن (Memory Location) کو دیا جانے والا نام ہے۔ جیسا کہ ڈیٹا کو کمپیوٹر کی میموری میں محفوظ کیا جاتا ہے۔ متغیر کی قیمت پروگرام میں بدلی جاسکتی ہے۔ اس سے مراد یہ ہے کہ اگر ایک متغیر کی قیمت 5 ہے تو ہم بعد میں اس کی قیمت 5 کی جگہ کچھ اور رکھ سکتے ہیں۔

ہر متغیر کا ایک منفرد نام ہوتا ہے جسے شناخت کنندہ **(Identifier)** کہتے ہیں اور ایک ڈیٹا ٹائپ **(Data Type)** ہوتی ہے۔ ڈیٹا ٹائپ بتاتی ہے کہ متغیر میں کس قسم کا ڈیٹا محفوظ کیا جاسکتا ہے۔ C- لینگوئج میں مختلف ڈیٹا ٹائپس ہیں جیسے char,float,int بالترتیب انٹیجر، حقیقی اور کریکٹر ڈیٹا کو محفوظ کرنے کے لیے استعمال ہوتی ہیں۔ ٹیبل **1.2** میں ڈیٹا کی مختلف اقسام کے مطابق C لینگوئج کی ڈیٹا ٹائپس بتائی گئی ہیں۔

| **Type of Data** | **Matching Data Type in C language** | **Sample Values** |
|---|---|---|
| Integer | *int* | 123 |
| Real | *float* | 23.5 |
| Character | *char* | 'a' |

ٹیبل **1.2**: مختلف قسم کے مستقل اعداد کی ڈیٹا ٹائپس

اب ہم ممکنہ ڈیٹا ٹائپس اور متغیّرات کے ناموں کے بارے میں تفصیل سے پڑھیں گے۔

### 1.3.3 متغیر کی ڈیٹا ٹائپ (Data Type of a Variable)

C- لینگوئج میں ہر متغیر کی ایک ڈیٹا ٹائپ ہوتی ہے۔ ڈیٹا ٹائپ نہ صرف یہ بتاتی ہے کہ اس متغیر میں کس قسم کا ڈیٹا محفوظ کیا جاسکتا ہے بلکہ یہ بھی بتاتی ہے کہ اس متغیر کا ڈیٹا محفوظ کرنے کے لیے کمپائلر کو کتنی بائٹس رکھنی چاہییں ۔ نیچے C- لینگوئج کی مختلف ڈیٹا ٹائپس دی گئی ہیں۔

**کیا آپ جانتے تھے؟**

کچھ کمپائلر انٹیجر کو محفوظ کرنے کے لیے 2 بائٹ استعمال کرتے ہیں۔ ایسے کمپائلرز میں انٹیجر کی حد -32,768 سے 32،768 ہوتی ہے۔

### انٹیجر int۔ (Signed/Unsigned)

انٹیجر ڈیٹا ٹائپ انٹیجر کانسٹینٹس کو محفوظ کرنے کے لیے ہوتی ہے۔ انٹیجر کے لیے میموری کی 4 بائٹس درکار ہوتی ہیں۔ انٹیجر ٹائپ کا متغیر ڈکلیئر کرنے کے لیے ہم کی۔ورڈ int استعمال کرتے ہیں۔

**: Signed int**

Signed int میں -2,147,483,648 سے 2147,483,647 تک تمام مثبت اور منفی قیمتیں محفوظ کی جاسکتی ہیں۔ اگر صرف int ٹائپ دی جائے تو یہ طے شدہ ہے کہ اسے Signed int ہی سمجھا جائے گا۔

**:Unsigned int**

Unsigned int میں صرف مثبت قیمتیں ہی آسکتی ہیں جو 0 سے 4,294,967,295 تک ہوسکتی ہیں۔ ان سائنڈ انٹیجر ڈکلیئر کرنے کے لیے کی۔ورڈ unsigned int استعمال ہوتا ہے۔

### فلوٹنگ پوائنٹ float:

فلوٹ ڈیٹا ٹائپ میں حقیقی نمبر آتے ہیں جن میں اعشاریہ کے بعد زیادہ سے زیادہ چھ ہندسے آسکتے ہیں۔ فلوٹ ٹائپ کا متغیر ڈکلیئر کرنے کے لیے کی۔ورڈ float استعمال ہوتا ہے۔ ایک فلوٹ میموری کی 4 بائٹس لیتا ہے۔ اس کی قیمتیں $3.4x10^{-38}$ سے $3.4x10^{38}$ تک ہوسکتی ہیں۔

### کریکٹر (char):

C-میں کریکٹر ٹائپ کے ویری ایبلز کو ڈکلیئر کرنے کے لیے کی۔ورڈ char کی ضرورت ہوتی ہے۔ اسے محفوظ کرنے کے لیے میموری کی 1 بائٹ درکار ہوتی ہے۔ char ٹائپ کے ایک ویری ایبل میں صرف ایک کریکٹر محفوظ کیا جاسکتا ہے۔

## 1.3.4 متغیرات کا نام (Name of a variable)

ہر متغیر کا ایک مخصوص نام یا شناخت کنندہ ہوتا ہے۔ متغیر کا نام رکھنے کے اصول درج ذیل ہیں:

1) متغیر کے نام میں صرف حروف (چھوٹے یا بڑے) ہندسے اور "_" علامت آسکتی ہے۔
2) متغیر کا نام کسی حرف یا "_" علامت سے شروع ہوسکتا ہے ہندسے سے نہیں۔
3) ایک کی۔ورڈ متغیر کا نام نہیں ہوسکتا۔

4) متغیر کے نام کی لمبائی کا کوئی اصول نہیں ہے لیکن بہتر یہ ہے کہ نام مختصر ہو۔

متغیرات کے درست ناموں کی کچھ مثالیں ہیں height، Average weight، _var1

**سرگرمی 1.7:**

درج ذیل میں سے متغیرات کے درست ناموں کے گرد دائرہ لگائیں۔

| | | | | |
|---|---|---|---|---|
| _Hello, | 1var | roll_num | Air23Blue | float |
| Case | $car | name | =color | Float |

**اہم نوٹ:**

پروگرامنگ کا عمدہ طریقہ یہ ہے کہ متغیرات کے نام ایسے ہونے چاہییں جیسا ڈیٹا اُن میں رکھنا ہو جیسا کہ اگر کسی متغیر میں تنخواہ محفوظ کرنی ہو اس کا نام salary یا wages ہونا چاہیے۔

### 1.3.5 متغیر کی ڈکلیریشن (Variable Declaration)

پروگرام میں ایک متغیر کو استعمال کرنے سے پہلے اسے ڈکلیئر کرنا ضروری ہے۔ متغیر کو ڈکلیئر کرتے ہوئے اس کی ڈیٹا ٹائپ اور نام بتایا جاتا ہے۔

متغیر کو ڈکلیئر کرنے کا سینٹیکس درج ذیل ہے:

***data_type variable_name;***

متغیرات کو ڈکلیئر کرنے کی چند مثالیں درج ذیل ہیں:

```
unsigned int age;

float height;

int salary;

char marital_status;
```

ایک سٹیٹمنٹ میں ایک سے زیادہ متغیرات کو بھی ڈکلیئر کیا جا سکتا ہے جیسے درج ذیل مثالوں میں کیا گیا ہے۔

```
unsigned int age, basic_salary, gross_salary;

int points_scored, steps;
```

```
float height, marks;
char marital_status, gender;
```

ڈیٹا ٹائپ بتائے بغیر ایک متغیر ڈکلیئر نہیں کیا جا سکتا۔ ڈکلیئر کرنے کے بعد اس متغیر کی ڈیٹا ٹائپ تبدیل نہیں کی جا سکتی۔ متغیر کو ڈکلیئر کرنے سے پتا چلتا ہے کہ اس کی قسم کیا ہے، اس میں کہاں سے کہاں تک قیمتیں آ سکتی ہیں اور اس پر کس قسم کے اوپریشنز انجام دیے جا سکتے ہیں درج ذیل مثال میں ایک پروگرام دیا گیا ہے، جس میں دو متغیرات ڈکلیئر کیے گئے ہیں۔

**EXAMPLE CODE 1.2**

```
void main()
{
        char grade;
        int value;
}
```

### 1.3.6 متغیر کی انشلائزیشن (Variable Initialization)

پہلی مرتبہ ایک متغیر سے قیمت کو منسوب کرنا متغیر کو انیشلائزیشن کہلاتا ہے۔ C- لینگویج میں متغیر کو ڈکلیئر کرتے ہوئے یا اس کے بعد انیشلائز کیا جا سکتا ہے۔ متغیر کو ڈکلیریشن کے وقت انیشلائز کرنے کے لیے عام سنٹیکس یہ ہے:

```
data_type  variable_name = value;
```

درج ذیل مثال میں ایک پروگرام دیا گیا ہے جس میں دو متغیرات کو ڈکلیر اور انیشلائز کیا گیا ہے۔

**EXAMPLE CODE 1.3**

```
#include<stdio.h>
void main()
{
        char grade; //Variable grade is declared
        int  value  =  25;  /*Variable  value  is  declared  and
        initialized.*/
        grade = 'A'; //Variable grade is initialized
}
```

**سرگرمی 1.8:**

ایک پروگرام لکھیں جو مناسب ڈیٹا ٹائپس کے متغیرات میں آپ کا ذاتی ڈیٹا محفوظ کرے۔

- نام کا پہلا حرف
- جنس کا پہلا حرف
- آپ کی عمر
- آپ کے آٹھویں جماعت کے نمبر
- آپ کی قامت

## خلاصہ:

- کمپیوٹر کو انسان ہدایات دیتے ہیں جن سے انھیں پتا چلتا ہے کہ ایک خاص مسئلے کو کیسے حل کرنا ہے۔ ہدایات کی اس فہرست کو کمپیوٹر پروگرام یا سافٹ ویئر کہا جاتا ہے۔

- کمپیوٹر کو ہدایات دینے یا محفوظ کرنے کا عمل کمپیوٹر پروگرامنگ کہلاتا ہے۔ اور وہ شخص جو جانتا ہو کہ ایک **کمپیوٹر پروگرام** کس طرح لکھا جاتا ہے پروگرامر کہلاتا ہے۔

- کمپیوٹر پروگرام جن زبانوں میں لکھے جاتے ہیں وہ پروگرامنگ لینگوئجز کہلاتی ہیں۔ عام طور پر جانی جانے والی چند پروگرامنگ لینگوئجز JAVA, PYTHON,C++, C ہیں۔

- پروگرامنگ کے لیے ضروری آلات کو اگر اکٹھا کر دیا جائے تو پروگرامنگ انوائرنمنٹ بنتی ہے۔ پروگرامنگ انوائرنمنٹ پروگرام لکھنے اور چلانے کے لیے بنیادی پلیٹ فارم فراہم کرتی ہے۔

- ایک سافٹ ویئر جو پروگرامر کو پروگرام لکھنے اور چلانے میں مدد دے **انٹیگریٹڈ ڈویلپمنٹ انوائرنمنٹ (IDE)** کہلاتا ہے۔

- ایڈیٹر یا ٹیکسٹ ایڈیٹر ایسا سافٹ ویئر ہوتا ہے جس میں پروگرامر پروگرام لکھ سکتا ہے اور اس میں ترمیم کر سکتا ہے۔ تمام IDE's کے اپنے مخصوص ایڈیٹرز ہوتے ہیں۔

- ہر پروگرامنگ لینگوئج میں چند ابتدائی تعمیراتی عناصر ہوتے ہیں اور یہ چند گرامر کے اصولوں کی پابند ہوتی ہے جنھیں سنٹیکس کہا جاتا ہے۔

- ہر پروگرامنگ لینگوئج میں پہلے سے ڈیفائن کیے ہوئے الفاظ کی ایک فہرست ہوتی ہے۔ اگر ایک پروگرامر انھیں کسی اور کام کے لیے استعمال کرنے کی کوشش کرے یا انھیں خود کوئی ڈیفینیشن دے تو سنٹیکس ایرر آتا ہے۔ یہ الفاظ کی ورڈز **(Keywords)** یا ذخیرہ الفاظ **(Reserved Words)** کہلاتے ہیں۔

- ایک پروگرام کو تین حصوں میں تقسیم کیا جاتا ہے۔ ہیڈر سیکشن وہ حصہ ہے جس میں ہیڈر فائلز شامل کی جاتی ہیں۔ مین سیکشن میں مین فنکشن آتا ہے۔ Main باڈی میں وہ سب کچھ آتا ہے جو { } قوسین میں لکھا گیا ہو۔

- کمنٹس وہ سٹیٹمنٹس ہوتی ہیں جنھیں کمپائلر نظر انداز کر دیتا ہے اور وہ چلتی نہیں ہیں۔ پروگرام کے بارے میں اضافی معلومات شامل کرنے کے لیے کمینٹس کو استعمال کیا جا سکتا ہے۔

- کانسٹینسیس کی قیمتیں تبدیل نہیں ہوتی۔ کانسٹینسیس کی تین اقسام انٹیجر کانسٹینس، رئیل کانسٹینس اور کریکٹر کانسٹینس ہیں۔

- ۔ ایک متغیر کو استعمال کرنے سے پہلے ڈکلیئر کرنا ضروری ہے۔ متغیر کی ڈکلیریشن کے دوران اس کا نام اور ڈیٹا ٹائپ بتائی جاتی ہے۔
- ۔ پہلی مرتبہ ایک متغیر سے قیمت منسوب کرنا، متغیر کی انشلائزیشن کہلاتا ہے۔ متغیر کو ڈکلریشن کے وقت یا اس کے بعد انشلائز کیا جا سکتا ہے۔

## مشق

### سوال نمبر 1: کثیر الانتخابی سوالات:

1) ایک سافٹ ویئر جو پروگرامر کو کمپیوٹر پروگرام لکھنے میں مدد دیتا ہے ________ کہلاتا ہے۔

a) کمپائلر    b) ایڈیٹر    c) IDE    d) ڈیبگر

2) ________ ایک ایسا سافٹ ویئر ہوتا ہے جو پروگرام کی فائلز کو ایسے کوڈ میں تبدیل کر دیتا ہے جسے مشین سمجھ سکے اور چلا سکے۔

a) کمپائلر    b) ایڈیٹر    c) IDE    d) ڈیبگر

3) ہر پروگرامنگ لینگوئج میں چند ابتدائی تعمیراتی عناصر ہوتے ہیں اور یہ گرامر کے چند اصولوں کے پابند ہوتے ہیں جنھیں ______ کہا جاتا ہے۔

a) پروگرامنگ رُولز    b) سنٹیکس    c) تعمیراتی عناصر    d) سیمانٹک رُولز

4) ایسے الفاظ کی فہرست جو پہلے سے ڈیفائنڈ ہیں اور جنھیں پروگرامر اپنے متغیرات کے ناموں کے طور پر استعمال نہیں کر سکتا ______ کہلاتے ہیں۔

a) آٹو ورڈز    b) کی۔ورڈز    c) محدود الفاظ    d) پہلے سے ڈیفائن کیے ہوئے الفاظ

5) include سٹیٹمنٹس __________ سیکشن میں لکھی جاتی ہیں۔

a) ہیڈر    b) مین(Main)    c) کیمنٹس    d) پرنٹ

6) ________ کو سورس(source) کوڈ میں پروگرام کے استعمال کیے ہوئے الگورتھم اور طریقہ کار کی مزید وضاحت کرنے کے لیے استعمال کیا جاتا ہے۔

a) پیغامات    b) اشارات    c) کمینٹس    d) وضاحتیں

7) ________ وہ قیمتیں جو پروگرام کے چلتے ہوئے تبدیل نہیں ہوتی۔

a) متغیرات    b) کانسٹینٹس    c) سٹرنگز    d) کمینٹس

8) ایک فلوٹ میموری کی ________ بائٹس استعمال کرتا ہے۔

a) 3    b) 4    c) 5    d) 6

9) ایک متغیر کو انیشلائز کرنے کے لیے ہم ________ اوپریٹر استعمال کرتے ہیں۔

a) -->    b) =    c) @    d) ?

10) ________ کو کانسٹینٹس محفوظ کرنے کے لیے ایک مرتبان سمجھا جا سکتا ہے۔

a) باکس    b) جار    c) متغیر    d) مجموعہ

## سوال نمبر 2 : غلط/ درست کی نشاندہی کریں۔

1) ایک IDE ٹیکسٹ ایڈیٹرز، کمپائلرز اور ڈیبگرز کو ایک انٹرفیس میں اکٹھا کرتی ہے۔ غلط/ درست
2) کمپیوٹر میں پروگرام فائل میں لکھے ہوئے کوڈ کو چلانے کے لیے بائنری لینگویج میں تبدیل کرنا پڑتا ہے۔ غلط/ درست
3) C- پروگرامنگ لینگویج میں column ایک ذخیرۂ الفاظ ہے۔ غلط/ درست
4) *comment goes here* ایک درست کمنٹ ہے۔ غلط/ درست
5) فلوٹ میں چھے ہندسوں تک پریسائز رئیل نمبر محفوظ کیا جاسکتا ہے۔ غلط/ درست

## سوال نمبر 3 : درج ذیل کی تعریف کریں۔

1) IDE 2) کمپائلر 3) کی۔ ورڈز 4) پروگرام کا مین سیکشن 5) Char ڈیٹا ٹائپ

## سوال نمبر 4 : درج ذیل سوالات کے مختصر جوابات تحریر کریں۔

1) ہمیں ایک پروگرامنگ انوائرنمنٹ کی ضرورت کیوں ہوتی ہے؟
2) اپنی لیب کے کمپیوٹر میں موجود IDE میں C- پروگرام فائل بنانے کے مراحل لکھیں؟
3) کمپائلر کے مقصد کی وضاحت کریں؟
4) C- پروگرامنگ لینگویج کے پانچ کی۔ ورڈز کی فہرست تحریر کریں؟
5) C- پروگرام کی ساخت کے اہم حصےّ بتائیں؟
6) پروگرامنگ میں کمینٹس کیوں استعمال کرتے ہیں؟
7) کانسٹینٹس اور متغیرات میں فرق کریں۔
8) متغیرات کے نام رکھنے کے اصول تحریر کریں۔
9) char اور int میں فرق بتائیں۔
10) ہم ایک متغیر کو کس طرح ڈکلیئر اور انیشلا ئز کر سکتے ہیں۔

## سوال نمبر 5 : کالم ملائیں۔

| A | B | C |
|---|---|---|
| 1) IDE | a) ایسا کوڈ جو مشین پہ چلایا جاسکے | |
| 2) ایڈیٹر | b) Include سٹیٹمنٹ | |
| 3) کمپائلر | c) Pythone | |
| 4) پروگرامنگ لینگویج | d) CLion | |
| 5) ذخیرۂ الفاظ | e) /*(a+b)*/ | |
| 6) لنک سیکشن | f) Notepad | |
| 7) Main() کی باڈی | g) Int weight | |

| | | |
|---|---|---|
| 8) کمنٹ | h) struct | |
| 9) انٹیجر متغیر | i) 4 بائیٹس | |
| 10) فلوٹ | j) { } | |

## پروگرامنگ کی مشقیں

**مشق نمبر 1:**

- استادِ محترم کی مدد سے اپنی لیب کے کمپیوٹر پر C- پروگرام لکھنے کے لیے انسٹال کی ہوئی IDE کھولیں۔
- ایڈیٹر میں یہ پروگرام لکھیں اور welcome نام رکھ کر محفوظ (save) کر لیں۔

```
#include <stdio.h>
#include <conio.h>
void main()
{
        /*A simple C language program*/
        printf(“Welcome to C language”);
        getch();
}
```

- سکرین پر "welcome to c language" بطور آؤٹ پٹ دیکھنے کے لیے پروگرام چلائیں۔

**مشق نمبر 2:**

ایک پروگرام لکھیں جو آپ کے بہترین دوست کا ذاتی ڈیٹا محفوظ کرنے کے لیے مناسب ڈیٹا ٹائپس کے متغیرات ڈکلیئر کرے۔ ان متغیرات کو درج ذیل ڈیٹا سے انیشلائز کریں۔

- اس کے نام کا پہلا حرف
- اس کی جنس کا پہلا حرف
- اس کی عمر
- اس کے قد کی لمبائی

باب 2

# یوزر انٹریکشن

## تدریسی مقاصد:(Students Learning Outcomes)

- آؤٹ پٹ فنکشن جیسے: Printf() کا استعمال
- ان پٹ فنکشن جیسے:
  - scanf( )
  - getch() کا استعمال
- سٹیٹمنٹ ٹرمینیٹر (سیمی کولن) کا استعمال
- فارمنٹ سپیسفائرز کی تعریف
  - انٹیجر (%i)
  - ڈیسیمل (%d)
  - فلوٹ (%f)
  - کریکٹر (char) (%c)
- اسکیپ سیکوئنس کی تعریف
- درج ذیل اسکیپ سیکوئنسز کی پروگرامنگ کی مثالوں کے ساتھ وضاحت:
  - نیولائن (/n)
  - ٹیب (/t)
- ارتھمیٹک اوپریٹر کی تعریف
- درج ذیل ارتھمیٹک اوپریٹرز کا استعمال
  - جمع (+)
  - تفریق (-)
  - ضرب (*)
  - تقسیم (/)
  - ریمنڈر/باقی (%)

# باب 2

- اسائنمنٹ اوپریٹرز کا استعمال
- ریشنل اوپریٹرز کی تعریف
- درج ذیل ریشنل اوپریٹرز کا استعمال
  - سے کم (<)
  - سے زیادہ (>)
  - سے کم یا برابر (<=)
  - سے زیادہ یا برابر (>=)
  - کے برابر (=)
  - کے برابر نہیں (!=)
- منطقی اوپریٹر کی تعریف
- درج ذیل منطقی اوپریٹر کا استعمال
  - AND (&&)
  - OR (||)
  - NOT (!)
- اسائنمنٹ اوپریٹر (=) اور برابر کے اوپریٹر (= =) میں فرق
- یونری اور بائنری اوپریٹرز میں فرق
- اوپریٹرز کی ترجیح کی ترتیب کی تعریف اور وضاحت

## تعارف (Introduction)

کمپیوٹر ایک ایسا آلہ ہے جو ڈیٹا کو بطور ان پٹ لیتا ہے اس ڈیٹا کو پروسیس کرتا ہے اور آؤٹ پٹ دیتا ہے۔ اسی لیے تمام پروگرامنگ کی زبانوں (Programming Language) میں ڈیٹا کی ان پٹ، آؤٹ پٹ اور پروسیسنگ کے لیے ہدایات ضروری جاتی ہیں ۔ اس باب میں ہم C- لینگوئج (C-Language) کے مختلف پری بِلٹ (Pre-Built) ان پٹ، آؤٹ پٹ فنکشنز (Funcitons) کے بارے میں بھی پڑھیں گے۔ ہم ڈیٹا کو پروسیس (Process) کرنے کے لیے استعمال ہونے والے مختلف اوپریٹرز (Operators) کے بارے میں پڑھیں گے۔

## 1.2 اِن پٹ/آؤٹ پٹ (I/O) فنکشن (Input/Output Functions):

ہمیں پروگرام لکھتے ہوئے ان پٹ دینے اور آؤٹ دکھانے کے لیے ایک لائحہ عمل درکار ہوتا ہے۔ ان پٹ/آؤٹ پٹ آپریشنز کے لیے ہر پروگرامنگ لینگوئج کے اپنے کی-ورڈز (Keywords) یا سٹینڈرڈ لائبریری فنکشنز (Standard Library Functions) ہوتے ہیں۔ C- لینگوئج (C-Language) آؤٹ پُٹ دکھانے کے لیے printf فنکشنز اور صارف (User) سے ان پٹ لینے کے لیے scanf فنکشن مہیا کرتی ہے۔ اس سیکشن (Section) میں ہم یہ دو فنکشن پڑھیں گے۔

### 1.1.2 Printf():

Printf C- پروگرامنگ لینگوئج میں سکرین پر آؤٹ پٹ دکھانے کا ایک بلٹ اِن (Built-in) فنکشن ہے. اس کا نام پرنٹ فارمیٹڈ (Print Formatted) سے نکلا ہے کیوں کہ یہ سکرین پر فارمیٹ کی ہوئی آؤٹ پٹ دکھانے کے لیے استعمال ہوتا ہے۔ پچھلے باب میں ڈیٹا کی جن اقسام کا ذکر کیا گیا ہے وہ سب printf فنکشن سے ڈسپلے کی جا سکتی ہیں مثال کے طور پر نیچے دیے گئے پروگرام کو دیکھیں۔

**EXAMPLE CODE 2.1 <\>**

```
#include<stdio.h>
void main ()
{
      printf(“Hello World”);
}
Output:
Hello World
```

اس مثال میں سکرین پر Hello World دکھانے کے لیے printf فنکشن استعمال کیا گیا ہے۔ ہم واوین ("  ") کے درمیان میں جو بھی لکھتے ہیں وہ سکرین پر آجاتا ہے۔

**سرگرمی 2.1:**

دیے گئے کوڈ کی آؤٹ پٹ لکھیں۔

```
# include<stdio.h>
void main()
{
      printf(“I am UPPERCASE and this is l owercase”);
}
```

**سرگرمی 2.2:**

سکرین پر اپنے نام کا پہلا حصّہ اپر کیس (Upper Case) اور دوسرا حصّہ لوئر کیس (Lower Case) میں دکھانے کے لیے پروگرام لکھیں۔

## 2.1.2 فارمیٹ سپیسفائر (Fromate Specifier)

اگر ہم ایک ویری ایبل کی قیمت دکھانا چاہیں تو آیئے! ہم ایک ویری ایبل ڈکلیئر کرتے ہیں اور پھر دیکھتے ہیں printf کیسے کام کرتا ہے۔

```
int age = 35;
```

اب ہم اس ویری ایبل کی قیمت سکرین پر دکھانا چاہتے ہوں تو ہم سٹیٹمنٹ (Statement) لکھیں گے۔

```
printf("age");
```

لیکن اس سے ہمارا مقصد حاصل نہیں ہوا کیونکہ یہ سکرین پر دکھاتا ہے۔

```
age
```

یہ ویری ایبل age میں محفوظ کی ہوئی قیمت نہیں دکھاتا بلکہ جو بھی printf کی واوین میں لکھا ہے وہ ہی دکھاتا ہے۔ درحقیقت ہمیں فارمیٹ سپیسفائر کو استعمال کرتے ہوئے اس ڈیٹا کے فارمیٹ کی وضاحت کرنی پڑے گی جو ہم سکرین پر دکھانا چاہتے ہیں۔ C-لینگوئج میں ڈیٹا کی مختلف اقسام کے لیے جو فارمیٹ سپیسفائر ہیں وہ ٹیبل 2.1 میں دیے گئے ہیں۔

| Data Type | Format Specifier |
|---|---|
| int | % d or % i |
| float | % f |
| char | % c |

Table 2.1: ان پٹ/آؤٹ پٹ آپریشنز کے لئے فارمیٹ سپیسفائر

فرض کریں ہم int ٹائپ کا ڈیٹا دکھانا چاہتے ہیں تو ہمیں فارمیٹ سپیسفائر %d یا %i کو استعمال کرتے ہوئے printf میں اس کی وضاحت کرنی ہوگی۔ اسی طرح سے float ٹائپ کے لئے ہم %f استعمال کریں گے۔ یہ درج ذیل مثال میں واضح کیا گیا ہے۔

## EXAMPLE CODE 2.2

```
#include<stdio.h>
void main()
{
        float height = 5.8;
        int age = 35;
        printf(“My age is %d and my height is %f”, age, height);
}
```

**Output:**

```
My age is 35 and my height is 5.800000
```

اوپر دی گئی مثال میں ہم دیکھ سکتے ہیں کہ آؤٹ پٹ دکھاتے ہوئے پہلے فارمیٹ سپیسفائر کی جگہ واوین کے بعد دیے گئے ڈیٹا میں سے پہلے ویری ایبل یعنی age کی قیمت آ گئی ہے اور دوسرے فارمیٹ سپیسفائر کی جگہ دوسرا ویری ایبل آ گیا ہے۔

**اہم معلومات:**

جب ہم float کی قیمت دکھانے کے لیے %f استعمال کرتے ہیں تو یہ ڈیسمل پوائنٹ (Decimal Point) کے بعد چھ ہندسے دکھاتا ہے۔ اگر ہم چاہتے ہیں کہ ڈیسمل پوائنٹ کے بعد آنے والے ہندسوں کی تعداد خود بتائیں تو ہم %nf لکھ سکتے ہیں جس میں n سے مراد ہندسوں کی تعداد ہے۔ اوپر دی گئی مثال میں اگر ہم یہ سٹیٹمنٹ لکھیں۔

```
printf(“My age is %d and my height is % 2f”, age, height);
```

تو یہ آوٹ پٹ ہوگی۔

```
My age is 35 and my height is 5.80
```

**اہم نوٹ:**

فارمیٹ سپیسفائر صرف ویری ایبل کے لیے استعمال نہیں کیے جاتے ہیں دراصل وہ ہر اس ایکسپریشن (Expression) کا نتیجہ دکھانے کے لئے استعمال ہوتے ہیں جس میں ویری ایبل ہوں، کانسٹینٹس ہوں یا دونوں ہوں۔ جیسا کہ اس مثال سے واضح کیا گیا ہے۔

**EXAMPLE CODE 2.3**

```
# include <stdio.h>
void main ()
{
        printf(“Sum of 23 and 45 is  %d”, 23 + 45);
}
```

**Output**

```
Sum of 23 and 45 is 68
```

### 2.1.3 Scanf()

scanf C- لینگویج میں ایک بلٹ -اِن فنکشن (Built-in Function) ہے جو صارف سے ویری ایبلز میں ان پٹ لینے کے لیے استعمال ہوتا ہے۔ scanf فنکشن میں ہم فارمیٹ سپیسفائر کی مدد سے ان پٹ ڈیٹا کی متوقع قسم کی وضاحت کرتے ہیں۔ اگر صارف int ٹائپ کا ڈیٹا دیتا ہے تو scanf میں لکھا گیا فارمیٹ سپیسفائر d% یا i% ہونا چاہیے۔ درج ذیل مثال پر غور کریں۔

**EXAMPLE CODE 2.4**

```
#include <stdio.h>
void main ()
{
        char grade;
        scanf(“%c”, &grade);
}
```

اس مثال میں فارمیٹ سپیسفائر c% کریکٹر ٹائپ کے ویری ایبل کے لیے استعمال کیا گیا ہے۔ صارف کی دی گئی ان پٹ ویری ایبل Grade میں محفوظ کی گئی ہے۔ scanf فنکشن کے دو اہم حصے ہیں جیسا کہ اوپر دیے گئے کوڈ میں دیکھا جا سکتا ہے۔ پہلے حصے میں واوین کے اندر فارمیٹ سپیسفائر کی ایک فہرست ہوتی ہے اور دوسرے حصے میں ویری ایبلز کی فہرست ہوتی ہے۔ جس کے بائیں طرف & لکھا جاتا ہے۔

**EXAMPLE CODE 2.5**

```
# include <stdio.h>
void main ()
{
        int number;
        printf(“Enter a number between 0-10: ”);
        scanf(“%d”, &number);
        printf(“The number you entered is: %d”, number);
}
```

**Output:**

```
Enter a number between 0-10: 4
The number you entered is: 4
```

**اہم نوٹ:**

ہم ایک scanf فنکشن کو استعمال کرتے ہوئے ایک سے زیادہ ان پٹس لے سکتے ہیں۔ مثال کے طور پر اس سٹیٹمنٹ کو دیکھیں۔

`scanf("%d%d%f",&a,&b,&c);`

یہ سٹیٹمنٹ دو انٹیجر (integer) ٹائپ کے ویری ایبلز a اور b اور ایک float ٹائپ کے ویری ایبل میں ان پٹ لیتی ہے۔ ہر ان پٹ کے بعد صارف کو space یا enter دبانا ہوتا ہے اور تمام ان پٹس دینے کے بعد enter دبانا لازمی ہوتا ہے۔

**سرگرمی 2.3**

ایک پروگرام لکھیں جو صارف سے رول نمبر، نمبروں کی پرسنٹیج اور گریڈ بطور ان پٹ لے اور اس طرح سے فارمیٹ کی ہوئی آؤٹ پٹ دکھائے۔

```
Roll No     :        input value
Percentage  :        input value %
Grade       :        input value
```

**اہم نوٹ:**

scanf فنکشن میں & اوپریٹر کو بھول جانا بہت عام غلطی ہے۔ & اوپریٹر کے بغیر پروگرام چل تو جاتا ہے پر توقع کے مطابق نہیں چلتا۔

### 2.1.4: getch()

getch() فنکشن صارف سے ایک کریکٹر لینے کے لیے استعمال ہوتا ہے۔ صارف جو کریکٹر ان پُٹ کرتا ہے وہ سکرین پر نہیں ظاہر ہوتا۔ یہ فنکشن عام طور پر پروگرام کی ایگزیکیوشن (execution) کو روکنے کے لیے استعمال کیا جاتا ہے کیونکہ پروگرام اس وقت تک آگے نہیں چلتا جب تک صارف کوئی بٹن نہیں دباتا۔ اس فنکشن کو استعمال کرنے کے لیے ہم پروگرام کے ہیڈر سیکشن (header section) میں ایک لائبریری conio.h شامل کرتے ہیں۔

**EXAMPLE CODE 2.6**

```
# include<stdio.h>
# include<conio.h>
void main ()
{
        printf(“Enter any key if you want to exit program ”);
        getch();
}
```

اوپر دی گئی مثال میں پروگرام صارف سے کریکٹر اِن پُٹ کرنے کا مطالبہ کرتا ہے اور پھر پروگرام ایگزیکیوشن ختم کرنے سے پہلے ان پٹ کا انتظار کرتا ہے۔

**EXAMPLE CODE 2.7**

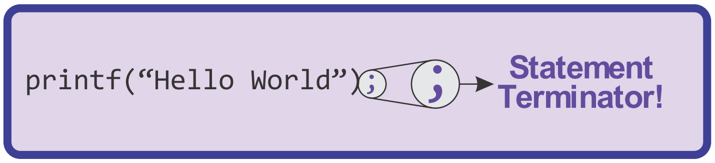
```
# include<stdio.h>
# include<conio.h>
void main ()
{
        char key;
        printf(“Enter any key : ”);
        key = getch(); //Gets a character from user into variable key
}
```

اگر ہم یہ پروگرام ایگزیکیوٹ کریں تو ہمیں scanf اور getch فنکشن سے کریکٹر لینے میں فرق کا پتا چلے گا۔ جب ہم scanf کے ذریعے کریکٹر اِن پُٹ کرتے ہیں تو ہمیں پروگرام کو آگے چلانے کے لئے enter دبانا پڑتا ہے۔ لیکن getch میں پروگرام enter دبانے کا انتظار نہیں کرتا۔ یہ فنکشن کریکٹر لیتے ہی اگلی سٹیٹمنٹ ایگزیکیوٹ کرنا شروع کر دیتا ہے۔

## 2.1.5 سٹیٹمنٹ ٹرمینیٹر (Statement Terminator)

سٹیٹمنٹ ٹرمینیٹر کمپائلر کو یہ بتاتا ہے کہ لائن ختم ہوگئی ہے C-لینگوئج میں سیمی کولن (;) بطور سٹیٹمنٹ ٹرمینیٹر استعمال ہوتا ہے۔

اگر ہم ہر سٹیٹمنٹ کے آخر میں ";" نہ لگائیں تو نتیجتاً error آجاتا ہے۔



```
printf(“Hello World”);
```

## 2.1.6 اسکیپ سیکوئنس (Escape Sequence)

**مقصد:**

اسکیپ سیکوئنس printf فنکشن میں واوین کے درمیان استعمال ہوتے ہیں ان کے ذریعے printf فنکشن معمول کے طریقہ کار سے ہٹ کر آؤٹ پٹ دکھاتا ہے اس کو سمجھنے کے لئے ہم نیچے دی گئی سٹیٹمنٹ کو دیکھتے ہیں۔

```
printf("My name is\"Ali\"");
```

اوپر دی گئی سٹیٹمنٹ کی آؤٹ پٹ یہ ہے۔

```
My name is "Ali"
```

اوپر دی گئی مثال میں "\" ایک اسکیپ سیکوئنس ہے یہی وجہ ہے کہ printf "" کو سکرین پر دکھاتا ہے۔

**اسکیپ سیکوئنس کی بناوٹ:**

اسکیپ سیکوئنس میں دو کریکٹر ہوتے ہیں پہلا کریکٹر ہمیشہ بیک سلیش (back slash) "\" ہوتا ہے اور دوسرا کریکٹر مطلوبہ فنکشنیلٹی کے مطابق آتا ہے ۔ بیک سلیش (\) کو اسکیپ کریکٹر (escape character) کہا جاتا ہے کیونکہ یہ ہر اسکیپ سیکوئنس کے ساتھ اسکیپ کی نشان ہی کرنے کے لیے منسلک ہوتا ہے۔ اسکیپ کریکٹر اور اس کے بعد آنے والا کریکٹر سکرین پر نہیں لکھے جاتے لیکن یہ وہ مخصوص کام کرتے ہیں جو ان سے منسوب کیا گیا ہے۔

**کیا آپ جانتے تھے؟**

اس سیکشن میں جن اسکیپ سیکوئنسز کا ذکر کیا گیا ان کے علاوہ C-لینگوئج کے درج ذیل اسکیپ سیکوئنس بھی عموماً استعمال ہوتے ہیں۔

| سیکوئنس | مقصد | سیکوئنس | مقصد |
|---|---|---|---|
| \' | سنگل کوٹ سکرین پر دکھانا (') | \a | الرٹ کی آواز پیدا کرتا ہے |
| \\ | بیک سلیش سکرین پر دکھانا (\) | \b | پچھلا کریکٹر مٹاتا ہے |

### نیولائین "\n"(New Line)

اسکیپ کریکٹر کے بعد n واضح کرتا ہے کہ کرسر (Cursor) کو اگلی لائن کے شروع پر لے کر جانا ہے۔ یہ اسکیپ سیکوئنس آؤٹ پٹ کو ایک سے زیادہ لائنوں میں پرنٹ کرنے کے لیے استعمال ہوتا ہے۔ اسکیپ سیکوئنس کو مزید سمجھنے کے لیے اس مثال پر غور کریں۔

**EXAMPLE CODE 2.8**

```
#include <stdio.h>
void main ()
{
        printf(“My name is Ali. \n”);
        printf(“I live in Lahore.”);
}
```

**Output**

```
My name is Ali.
I live in Lahore.
```

**اہم نوٹ:**

اسکیپ سیکوئنس کے بغیر اگر ہمارے پاس ایک سے زیادہ printf سٹیٹمنٹس بھی ہوں تو آؤٹ پٹ ایک ہی لائن پر آئے گی۔ اس مثال سے یہ بات واضح ہوتی ہے۔

**EXAMPLE CODE 2.9**

```
#include<stdio.h>
void main()
{
        printf(“My name is”);
        printf(“ Ahmad”);
}
```

**Output**

```
My name is Ahmad
```

### ٹیب "\t"(Tab)

اسکیپ سیکوئنس \t ,I/O فنکشن کو بتاتا ہے کہ افقی طور پر اگلے ٹیب سٹاپ (Tab Stop) پر جانا ہے۔ ایک ٹیب سٹاپ آٹھ سپیسز کا مجموعہ ہوتا ہے \t استعمال کرنے سے کرسر اگلے ٹیب سٹاپ پر چلا جاتا ہے۔

**EXAMPLE CODE 2.10**

```
#include<stdio.h>
void main ()
{
        printf("Name: \tAli\nFname: \tHammad\nMarks: \t1000");
}
```

**Output**

```
Name:   Ali
Fname:  Hammad
Marks:  1000
```

## 2.2 اوپریٹرز (Operators)

کمپیوٹر کے نام سے واضح ہوتا ہے کہ اس کا سب سے اہم کام کمپیوٹیشن (Computation) کرنا ہے۔ ہم پروگرامنگ کے ذریعے ڈیٹا پر کمپیوٹیشنز کرتے ہیں۔ ڈیٹا پر حساب کتاب کرنے کے لیے ہمارے پاس بہت سے ریاضی کے فنکشنز ہیں۔ ہم اپنے پروگراموں میں ریاضی کے آپریشنز بھی استعمال کر سکتے ہیں۔ C- لینگویج میں ڈیٹا کو ترتیب دینے اور پروسس کرنے کے لیے بہت سے آپریٹرز ہیں۔ کچھ بنیادی آپریٹرز کی فہرست درج ذیل ہیں:

- اسائنمنٹ آپریٹرز (Assignment Operators)
- ریلیشنل آپریٹرز (Relational Operators)
- ارتھمیٹک آپریٹرز (Arithmatic Operators)
- منطقی آپریٹرز (Logical Operators)

### 1.2.2 اسائنمنٹ آپریٹرز (Assignment Operators)

اسائنمنٹ آپریٹر ویری ایبل کو قیمت تفویض (Assign) کرنے کے لیے یا ایک ویری ایبل کی قیمت دوسرے ویری ایبل میں رکھنے کے لیے استعمال ہوتا ہے۔ C- لینگویج میں برابری کی علامت (=) بطور اسائنمنٹ آپریٹرز استعمال ہوتا ہے۔

C- لینگویج میں "=" کا سمبل اسائنمنٹ اوپریٹر کے لیے استعمال کیا جاتا ہے۔ مثال کے طور پر:

```
int sum=5;
```

کوڈ کی یہ لائن اگزیکیوٹ ہونے کے بعد ویری ایبل sum میں قیمت 5 محفوظ ہوجائے گی۔ اب ایک مثال دیکھتے ہیں۔

```
int sum=6;
int var=sum;
```

پہلے ویری ایبل sum کی قیمت 6 رکھی پھر اگلی لائن میں sum کی قیمت ویری ایبل var سے منسوب کر دی۔

### 2.1 پروگرامنگ ٹائم (Programming Time)

ایک پروگرام لکھیں جو دو انٹیجر متغیرات کی قیمتوں کا آپس میں تبادلہ کر دے۔

```
void main()
{
        int a = 2, b = 3, temp;
        temp = a;
        a = b;
        b = temp;
        printf("Value of a after swapping: %d\n", a);
        printf("Value of b after swapping: %d\n", b);
}
```

## 2.2.2 ارتھمیٹک آپریٹرز (Arithmatic Operators)

ارتھمیٹک آپریٹرز ڈیٹا پر ارتھمیٹک آپریشنز انجام دینے کے لیے استعمال ہوتے ہیں۔ ٹیبل 2.2 میں ارتھمیٹک آپریٹرز اور ان کی تفصیل دی گئی ہے:

| اوپریٹرز | نام | تفصیل |
|---|---|---|
| / | تقسیم کا آپریٹر | یہ بائیں طرف کی قیمت کو دائیں طرف والی پر تقسیم کرتا ہے۔ |
| * | ضرب کا آپریٹر | یہ دو قیمتوں کو ضرب کرتا ہے |
| + | جمع کا آپریٹر | یہ دو قیمتوں کو جمع کرتا ہے |
| - | تفریق کا آپریٹر | یہ بائیں طرف کی قیمت سے دائیں طرف والی قیمت تفریق کرتا ہے |
| % | ماڈولس کا آپریٹر | یہ بائیں طرف کی قیمت کو دائیں طرف کی قیمت پر عددی تقسیم کرنے کے بعد بچنے والی رقم دیتا ہے۔ |

ٹیبل 2.2 میں ارتھمیٹک آپریٹرز

### تقسیم (Division):

تقسیم کا اوپریٹر بائیں اوپرینڈ (Operand) کی قیمت کو دائیں آوپرینڈ کی قیمت پر تقسیم کرتا ہے۔ مثال کے طور پر اس سٹیٹمنٹ کو دیکھیں۔

اس سٹیٹمنٹ کے بعد ویری ایبل result کی قیمت 1.5 ہے۔

```
float result=3.0/2.0;
```

**اہم نوٹ:**

اگر دونوں اوپرینڈ انٹیجر (int) ٹائپ کے ہوں گے تو تقسیم کا جواب بھی int ٹائپ کا ہوگا۔ جواب انٹیجر دینے کے لیے "باقی" کو نظر انداز کر دیا جاتا ہے کوڈ کی اس لائن کو دیکھیں۔

```
float result = 3.0 / 2;
```

اگر دونوں قیمتیں int ٹائپ کی ہوں گی تو جواب بھی انٹیجر ہوگا یعنی 1- جب یہ قیمت (1) فلوٹ ٹائپ کے ویری ایبل result سے منسوب کی جائے تو یہ (1) فلوٹ ٹائپ میں تبدیل ہو جاتا ہے۔ اس لیے ویری ایبل result میں 1.0 قیمت محفوظ ہوتی ہے۔

اگر ہم چاہتے ہیں کے جواب درست اور جامع ہو تو ضروری ہے کہ کوئی اوپرینڈ فلوٹ ٹائپ کا ہی ہو۔ کوڈ کی اس لائن کو دیکھیں۔

```
float result = 3.0 / 2;
```

اوپر دی گئی مثال میں متغیر result میں قیمت 1.5 محفوظ ہے۔

### 2.2 پروگرامنگ ٹائم (Programming Time)

```
/*This program takes as input the price of a box of chocolates and
the total number of chocolates in the box. The program finds and
displays the price of one chocolate.*/
# include <stdio.h>
void main ()
{
     float box_price, num_of_chocolates, unit_price;
     printf (“Please enter the price of whole box of chocolates: ”);
     scanf(“%f”, &box_price);
     printf (“Please enter the number of chocolates in the box: ”);
     scanf(“%f”, &num_of_chocolates);
     unit_price = box_price / num_of_chocolates;
     printf(“The price of a single chocolate is %f”, unit_price);
}
```

**Output:**

```
Please enter the price of whole box of chocolates: 150
Please enter the number of chocolates in the box: 50
The price of a single chocolate is 3.000000
```

## ضرب (Multiplication):

ضرب کا آپریٹر (*) ایک بائنری آپریٹر ہے۔ جو دو اعداد کو ضرب کرتا ہے درج ذیل سٹیٹمنٹ دیکھیں:

```
int multiply= 5*5;
```

اس سٹیٹمنٹ کے چلنے کے بعد ویری ایبل multiply کی قیمت 25 ہے۔

### 2.3 پروگرامنگ ٹائم (Programming Time)

```
/* Following program takes as input the length and width of a
rectangle. Program calculates and displays the area of rectangle on
screen. */
# include<stdio.h>
void main ()
{
      float length, width, area;
      printf("Please enter the length of rectangle: ");
      scanf("%f", &length);
      printf("Please enter the width of rectangle: ");
      scanf("%f", &width);
      area = length * width;
      printf("Area of rectangle is : %f", area);
}
```

**Output**

```
Please enter the length of rectangle: 6.5
Please enter the length of rectangle: 3
Area of rectangle is : 19.500000
```

### سرگرمی: 2.4

ایک پروگرام لکھیں جو چوکور کے ایک طرف کی لمبائی ان پٹ لے اور اس کے رقبے کا حساب لگائے۔

## جمع (Addition):

جمع کا آپریٹر (+) دو آپرینڈز کو جمع کرتا ہے۔ اس سٹیٹمنٹ کو دیکھیں:

```
int add=10+10;
```

نتیجتاً ویری ایبل add کی قیمت 20 ہے۔

### 2.4 پروگرامنگ ٹائم (Programming Time)

```
/* This program takes marks of two subjects from user and displays
the sum of marks on console. */
#include <stdio.h>
void main ()
{
        int sum, math, science;
        printf(“Enter marks of Mathematics: ”);
        scanf(“%d”, &math);
        printf(“Enter marks of Science: ”);
        scanf(“%d”, &science);
        sum = math + science;
        printf(“Sum of marks is : %d”, sum);
}
```

**Output**

```
Enter marks of Mathematics:  90
Enter marks of Science: 80
Sum of marks is : 170
```

### سرگرمی : 2.5

ایک پروگرام لکھیں جو جار A اور B میں موجود گیندوں کی تعداد ان پٹ لے اور گیندوں کی کل تعداد سکرین پر دکھائے۔

**اہم معلومات:**

سٹیٹمنٹ ;a=a+1 ویری ایبل a کی قیمت میں 1 کا اضافہ کرنے کے لیے استعمال ہوتی ہے۔ C- لینگویج میں اس سٹیٹمنٹ کو ;++a یا ;a++ بھی لکھ سکتے ہیں اسی طرح ;--a یا ;a-- ویری ایبل a کی قیمت میں 1 کی کمی کرنے کے لیے استعمال ہوتا ہے۔

## تفریق (Subtraction):

تفریق کا آپریٹر بائیں آپرینڈ میں سے دائیں آپرینڈ کو تفریق کرتا ہے آیے! درج ذیل سٹیٹمنٹ کو دیکھتے ہیں۔

```
int result=20-15;
```

تفریق کے بعد ویری ایبل result کی قیمت 5 ہے۔

### سرگرمی 2.6:

ایک پروگرام لکھیں جو قمیص کی اصل قیمت اور ڈسکاؤنٹ پرسنٹیج صارف سے ان پٹ لے اور قمیص کی اصل قیمت، قیمت پہ کیا گیا ڈسکاؤنٹ اور ڈسکاؤنٹ کے بعد والی قیمت سکرین پر دکھائے۔

## ماڈولس اوپریٹر (Modulus Operator):

ماڈولس آپریٹر (%) بائیں آپرینڈ کو دائیں آپرینڈ پر تقسیم کرتا ہے اور تقسیم کے بعد بچنے والی "باقی" رقم لوٹاتا ہے۔ ماڈولس اوپریٹر انٹیجر ڈیٹا ٹائپس پر کام کرتا ہے۔

```
int remaining = 14 % 3;
```

اگر ہم 14 کو 3 پر تقسیم کریں تو "2 باقی" بچے گا اس لئے ویری اَیبل remaining کی قیمت 2 ہو گی۔

### 2.5 پروگرامنگ ٹائم (Programming Time)

```
/* This program finds and displays the right most digit of an input
number. */
#include <stdio.h>
void main()
{
      int num, digit;
      printf(“Enter a number: ”);
      scanf(“%d”, &num);
      digit = num % 10;
      printf(“Right most digit of number you entered is: %d”,
      digit);
}
```

**Output**

```
Enter a number: 789
Right most digit of number you entered is : 9
```

**سرگرمی 2.7:**

ایک پروگرام لکھیں جو صارف سے 2 ہندسوں والا نمبر اِن پٹ لے اور ایک ہندسے کو دوسرے ہندسے سے ضرب دے کر آؤٹ پُٹ دکھائے۔

**سرگرمی 2.8:**

ایک پروگرام لکھیں جو سیکنڈ ان پٹ لے اور ان کے برابر گھنٹے، منٹ اور سیکنڈ شمار کرے۔

**اہم نوٹ:**

C- لینگوئج میں ارتھمیٹک سٹیٹمنٹس لکھتے ہوئے ایک عام غلطی الجبرا کے روزمرہ کے اصولوں کو استعمال کرنا ہے۔ مثلاً 6*y کو 6y لکھنا اور x*x*x کو $x^3$ لکھنا وغیرہ۔ اس کے نتیجے میں کمپائلر ایرر (compiler error) آتا ہے۔

**سرگرمی 2.9:**

الجبرا کے ایکسپریشنز کو C- ایکسپریشنز میں تبدیل کریں۔

$$x = 6y + z$$

$$x = yz^3 + 3y$$

$$z = x + \frac{y^2}{3x}$$

$$z = (x - 2)^2 + 3y$$

$$y = (x + \frac{3z}{2}) + z^3 + \frac{x}{z}$$

## 2.2.3 ریلیشنل آپریٹرز (Relational Operators)

ریلیشنل آپریٹر دو قیمتوں کے تعلق کا تعین کرنے کے لیے ان کا موازنہ کرتے ہیں ریلیشنل آپریٹرز بتاتے ہیں کہ آیا دونوں قیمتیں برابر ہیں یا برابر نہیں ہیں۔ ایک قیمت دوسری سے بڑی ہے یا چھوٹی ہے۔ C- لینگوئج میں ہم ریلیشنل آپریٹرز کو نیومیرک (Numeric) اور Char ٹائپ کے ڈیٹا پر استعمال کر سکتے ہیں ٹیبل 2.3 میں C- لینگوئج کے ریلیشنل آپریٹر اور ان کی تفصیل دی گئی ہے۔

| ریلیشنل آپریٹر | تفصیل |
|---|---|
| == | کے برابر ہے |
| != | کے برابر نہیں ہے |
| > | سے زیادہ |
| < | سے کم |
| >= | سے زیادہ یا برابر |
| <= | سے کم یا برابر |

ٹیبل 2.3: بنیادی ریلیشنل آپریٹرز تفصیل کے ساتھ

ریلیشنل آپریٹرز دو آپرینڈز پر آپریشنز انجام دیتے ہیں اور نتیجہ ایک بولین ایکسپریشن (Boolean Expression) کی صورت میں دیتے ہیں یعنی (true or false) صحیح یا غلط۔ اگر صحیح (true) ہو تو قیمت 1 ہو گی اور اگر غلط (false) ہو تو قیمت 0 سے ظاہر کی جائے گی۔ یہ تصوّر اس ٹیبل **2.4** میں واضح کیا گیا ہے۔

| ریلیشنل ایکسپریشن | تفصیل | جواب |
|---|---|---|
| 5==5 | 5 برابر ہے 5 کے؟ | true |
| 5!=7 | 5 برابر نہیں ہے 7 کے؟ | true |
| 5>7 | 5 بڑا ہے 7 سے؟ | false |
| 5<7 | 5 چھوٹا ہے 7 سے؟ | true |
| 5>=5 | 5 بڑا ہے یا برابر ہے 5 کے؟ | true |
| 5<=4 | 5 چھوٹا ہے یا برابر ہے 4 کے؟ | false |

ٹیبل 2.4: ریلیشنل آپریٹرز کی مثال کے ساتھ وضاحت

### 2.2.4 اسائنمنٹ آپریٹر (=) اور برابر کا آپریٹر (==) (Assignment operator(=)&Equal to Operator(==))

C-لینگوئج میں (==) آپریٹر یہ چیک کرنے کے لئے استعمال ہوتا ہے کہ دو ایکسپریشن برابر ہیں یا نہیں جبکہ (=) آپریٹر دائیں طرف والے ایکسپریشن کو بائیں طرف والے ویری ایبل سے منسوب کرتا ہے۔ ڈبل ایکوئل (==) آپریٹر چیک کرتا ہے کہ دونوں طرف یہ آپرینڈز برابر ہیں یا نہیں۔ سنگل ایکوئل (Single Equal) آپریٹر (=) دائیں آپرینڈ کو بائیں طرف والے ویری ایبل سے منسوب کرتا ہے۔

**اہم نوٹ:**

ہم ریلیشنل ایکسپریشن کا نتیجہ دکھانے کے لئے printf فنکشن بھی استعمال کر سکتے ہیں۔ مثال کے طور پر یہ سٹیٹمنٹس دیکھیں۔

```
printf("%d",5==5);              //This statement dispay 1
printf("%d",5>7);           //This statement dispay 0
```

**سرگرمی 2.10:**

نیچے دی گئی ایکسپریشنز کا بولین رزلٹ بتائیں۔ جبکہ ویری ایبلز اور ان کی قیمتیں یہ ہیں x=3، y=7

(2+5)>y
x!=(y-4)
-1<x

(x+4)==y
(y/2)>=x
(x+3)<=20

## 2.2.5 منطقی آپریٹرز (Logical Operators)

لاجیکل آپریٹرز بولین ایکسپریشنز پر آپریشن سرانجام دیتے ہیں اور جواب بھی ایک بولین ایکسپریشن ہوتا ہے۔ جیسا کہ ہم پڑھ چکے ہیں کہ ریلیشنل آپریشن کا جواب بولین ایکسپریشن ہوتا ہے اس لیے لاجیکل آپریٹرز ایک سے زیادہ ریلیشنل ایکسپریشنز کی قیمت نکالنے کے لیے استعمال ہوتے ہیں۔ ٹیبل 2.5 میں C- لینگوئج کے منطقی آپریٹرز دیے گئے ہیں:

| آپریٹر | تفصیل |
|---|---|
| \|\| | منطقی OR |
| && | منطقی AND |
| ! | منطقی NOT |

ٹیبل 2.5 بنیادی لاجیکل آپریٹرز اور ان کی تفصیل

### AND آپریٹر (&&):

AND آپریٹر (&&) دو بولین ایکسپریشنز بطور آپرینڈ لیتا ہے اور جواب اسی صورت میں true ہوتا ہے اگر دونوں آپرینڈز true ہوں۔ اگر کوئی ایک بھی آپرینڈ false ہو تو جواب false ہوتا ہے۔ ٹیبل 2.6 میں AND آپریٹر کا ٹروتھ ٹیبل دیا گیا ہے۔

| Expression | Result |
|---|---|
| False && False | False |
| False && True | False |
| True && False | False |
| True && True | True |

ٹیبل 2.6: AND آپریٹر کا ٹروتھ ٹیبل

## OR آپریٹر (||):

OR آپریٹر بولین ایکسپریشن لیتا ہے اور جواب true ہوتا ہے اگر کوئی ایک آپرینڈ بھی true ہو۔ ٹیبل 2.7 میں OR آپریٹر کا ٹروتھ ٹیبل دیا گیا ہے۔

| **Expression** | **Result** |
|---|---|
| False \|\| False | False |
| False \|\| True | True |
| True \|\| False | True |
| True \|\| True | True |

ٹیبل 2.7: OR آپریٹر کا ٹروتھ ٹیبل

## NOT آپریٹر (!):

NOT آپریٹر بولین ایکسپریشن کی قیمت کو الٹ دیتا ہے یعنی اگر قیمت درست (true) ہو تو غلط (false) کر دیتا ہے اور غلط (false) ہو تو درست (true) کر دیتا ہے۔

| **Expression** | **Result** |
|---|---|
| !(True) | False |
| !(False) | True |

ٹیبل 2.8 NOT آپریٹر کا ٹروتھ ٹیبل ہے

## لاجیکل آپریٹر کی مثالیں:

ٹیبل 2.9 میں لاجیکل آپریٹر کا تصور مثالوں کی مدد سے واضح کیا گیا ہے۔

| لاجیکل ایکسپریشن | تفصیل | جواب |
|---|---|---|
| 3<4&&7>8 | 3 چھوٹا ہے 4 سے 7AND بڑا ہے 8 سے؟ | false |
| 3==4\|\|3>1 | 3 برابر ہے 4 کے 3 OR بڑا ہے ایک سے؟ | true |
| !(4>2\|\|2==2) | NOT(4 بڑا ہے 2 سے 2 OR برابر ہے 2 کے)؟ | false |
| 6<=6&&!(1>2) | 6 چھوٹا ہے یا برابر ہے 6 کے NOT AND(1 بڑا ہے 2 سے)؟ | true |
| 8>9\|\|!(1<=0) | 8 بڑا ہے 9 سے NOT OR(1 چھوٹا ہے یا برابر ہے o کے)؟ | true |

ٹیبل 2.9: لاجیکل آپریٹر کی مثال کے ساتھ وضاحت

**کیا آپ جانتے تھے؟**

C-لینگوئج شارٹ سرکٹ ایویلیوئیشن (Short circuit evaluation) کرتی ہے۔ اس کا مطلب ہے کہ:

**1۔** اگر AND او پریٹر کو حل کرتے ہوئے ایکسپریشن آپریٹر کی بائیں طرف والا حصہ false ہو تو پورا ایکسپریشن حل کیے بغیر جواب false ہو جاتا ہے۔

**2۔** اگر OR او پریٹر کو حل کرتے ہوئے ایکسپریشن آپریٹر کی بائیں طرف والا حصہ true ہو تو پورا ایکسپریشن حل کئے بغیر جواب true ہو جاتا ہے۔

**سرگرمی 2.11:**

فرض کریں دیے گئے ویری ایبلز کی قیمتیں ہیں x=4, y=7, z=8 جوابی ایکسپریشن بتائیں۔

```
x== 2   ||  y==8                     7>= y   &&  z<5
z>=5    ||  x<= -3                   y==7    &&  !(true)
x!=y    ||   y<5                     !(z>x)
```

## 2.2.6 یونری بمقابلہ بائنری آپریٹرز (Uniry vs Binary Operators)

وہ تمام آپریٹرز جن کا اس باب میں ذکر کیا گیا ہے۔ آپرینڈز کی تعداد کی بنا پر ان کی بنیادی دو قسمیں ہیں:

**1۔ یونری آپریٹرز:** یونری آپریٹرز کے لیے صرف ایک آپرینڈ درکار ہوتا ہے مثلاً لاجیکل NOT آپریٹر کا ایک ہی آپرینڈ ہوتا ہے۔ سائن آپریٹر (-) بھی یونری آپریٹر کی ایک مثال ہے۔ جیسے -5

**2۔ بائنری آپریٹرز:** بائنری آپریٹر کے لیے دو آپرینڈ درکار ہوتے ہیں مثلاً تمام ارتھمیٹک اور ریلیشنل آپریٹرز ہیں۔ && اور || لاجیکل آپریٹر بھی بائنری آپریٹرز ہیں۔

**کیا آپ جانتے تھے؟**

C-لینگوئج میں ایک ٹرنری آپریٹر (Ternary Operator) بھی ہوتا ہے جو تین آپرینڈز پر کام کرتا ہے۔

## 2.2.7 آپریٹرز کی ترجیح (Operators Precedence)

اگر ایک ایکسپریشن میں ایک سے زیادہ آپریٹرز ہوں تو سوال یہ پیدا ہوتا ہے کہ پہلے کسے حل کریں گے۔ اس مسئلے کو حل کرنے کے لیے آپریٹر کی ترجیح ٹیبل **2.10** میں دی گئی ہے۔ جس آپریٹر کی ترجیح زیادہ ہو وہ دوسرے آپریٹر سے پہلے حل کیا جاتا ہے۔ اگر ایک جیسی ترجیح ہو تو دائیں طرف والے آپریٹر سے پہلے بائیں طرف والا آپریٹر حل کریں گے۔

**Example:**

```
result = 18  / 2 * 3 + 7 % 3 + ( 5 * 4);      // evaluate ( )
result = 18  / 2 * 3 + 7 % 3 + 20;            // evaluate /
result = 9 * 3 + 7 % 3 + 20;                  // evaluate *
result = 27 + 7 % 3 + 20;                     // evaluate %
result = 27 + 1 + 20;                         // evaluate +
result = 28 + 20;                             // evaluate +
result = 48;
```

| Operator | Precedence |
|---|---|
| ( ) | 1 |
| ! | 2 |
| * , / , % | 3 |
| + , - | 4 |
| > , < , >= , <= | 5 |
| == , != | 6 |
| && | 7 |
| \|\| | 8 |
| = | 9 |

ٹیبل 2.10: آپریٹرز اور ان کی ترجیح

**سرگرمی 12.2:**

ان ایکسپریشنز کے جوابات لکھیں۔

**6 / (5 + 3)**
**7 + 3 * (12 + 2)**
**25 % 3 * 4**
**34 – 9 * 2 / (3 * 3)**
**18 / (15 – 3 * 2)**

## خلاصہ:

ہمیں پروگرام لکھتے ہوئے ان پٹ دینے اور آؤٹ پٹ دکھانے کے لیےایک طریقہ کار کی ضرورت ہوتی ہے۔ ان پٹ / آؤٹ پٹ آپریشنز کے لیے ہر پروگرامنگ لینگوئج کے اپنے کی۔ورڈز (Keywords) یا سٹینڈرڈ بلٹ اِن فنکشنز ہوتے ہیں۔

- **Printf** C-لینگوئج کا ایک بلٹ ان فنکشن ہے۔ اس کا نام پرنٹ فارمیٹڈ (Print Formatted) سے نکلا ہے اور یہ سکرین پر فارمیٹڈ آؤٹ پٹ دکھانے کے لیے استعمال ہوتا ہے۔
- **فارمیٹ سپیسفائرز** ان پٹ اور آؤٹ پٹ اوپریشنز میں ڈیٹا کا فارمیٹ بتانے کے لیے استعمال ہوتا ہے۔ فارمیٹ سپیسفائر سے پہلے ہمیشہ پرسنٹیج (%) سائن آتا ہے۔
- **scanf** C-لینگوئج کا ایک بلٹ ان فنکشن ہے جو صارف سے ان پٹ لیتا ہے۔
- **getch()** فنکشن صارف سے ایک کریکٹر اِن پٹ لینے کے لیے استعمال ہوتا ہے۔ اس فنکشن کو صرف کریکٹر ان پٹ دیا جا سکتا ہے۔ صارف جو کریکٹر درج کرتا ہے وہ سکرین پر نظر نہیں آتا۔
- **سٹیٹمنٹ ٹرمینیٹر** کمپائلر کو یہ بتاتا ہے کہ لائن ختم ہوگئی ہے۔ C-لینگوئج میں سیمی کولن (;) بطور سٹیٹمنٹ ٹرمینیٹر استعمال ہوتا ہے۔
- **اسکیپ سیکوئنس** printf کو بتاتا ہے کہ معمول سے ہٹ کر کام کرنا ہے۔ یہ اسکیپ کریکٹر اور ایک کریکٹر جو خاص فنکشنیلیٹی سے منسوب ہوتا ہے ان کا مجموعہ ہے۔
- **اسکیپ سیکوئنس** \n بتاتا ہے کہ کرسر کو نئی لائن کے شروع پر لے جانا ہے۔ یہ اسکیپ سیکوئنس آؤٹ پٹ کو ایک سے زیادہ لائنوں پر دکھانے کے لیے استعمال ہوتا ہے۔
- **اسکیپ سیکوئنس** \t بتاتا ہے کہ کرسر کو افقی طور پر اگلے ٹیب سٹاپ پر لے جانا ہے۔ ایک ٹیب سٹاپ آٹھ سپیسز کا مجموعہ ہوتا ہے۔
- **ارتھمیٹک آپریٹرز**، اسائنمنٹ آپریٹر، ریلیشنل آپریٹر اور لاجیکل آپریٹرز بنیادی آپریٹرز ہیں۔
- ارتھمیٹک آپریٹرز ارتھمیٹک فنکشنز کو حل کرنے کے لیے ڈیٹا پر کیلکولیشنز سرانجام دینے میں استعمال ہوتے ہیں۔ ارتھمیٹک اپریٹرز /، *، -، +، % ہیں۔
- **ماڈولس آپریٹر** ایک بائنری آپریٹر ہے جو بائیں آپرینڈ کو دائیں آپرینڈ پر تقسیم کرتا ہے اور تقسیم کے بعد بچنے والی رقم جواب ہوتا ہے۔ ماڈولس آپریٹر انٹیجر (integer) ڈیٹا ٹائپس پر کام کرتا ہے۔

- **ریلیشنل اوپریٹرز** دو قیمتوں کے تعلق کا تعین کرنے کے لیے ان کا موازنہ کرتے ہیں تمام ریلیشنل اوپریٹرز بائنری اوپریٹرز ہیں۔ اور یہ بائیں طرف سے دائیں طرف کام کرتے ہیں۔
- **لاجیکل اوپریٹرز** بولین ایکسپریشنز پر اوپریشن انجام دیتے ہیں اور جواب ایک بولین ویلیو ہوتی ہے۔
- لاجیکل **AND** اوپریٹر کا جواب true ہوتا ہے اگر اوپریٹر کے دونوں اطراف کے ایکسپریشنز true ہوں جبکہ لاجیکل OR اوپریٹر کا جواب true ہوتا ہے جب دونوں میں سے کوئی بھی ایکسپریشن true ہو۔
- لاجیکل **NOT** اوپریٹر کا جواب true ہوتا ہے اگر ایکسپریشن false ہو اور جواب false ہوتا ہے اگر ایکسپریشن true ہو۔
- پورے ایکسپریشن کو حل کیے بغیر آپریشن کا جواب دینا شارٹ سرکٹنگ (Short circuiting) کہلاتا ہے۔
- اوپریٹرز تین قسم کے ہوتے ہیں یونری اوپریٹر، بائنری اوپریٹر اور ٹرنری اوپریٹر جن کو بالترتیب ایک، دو اور تین اوپرینڈز اوپریشن انجام دینے کے لیے درکار ہوتے ہیں۔
- اوپریٹرز کی ترجیح سے پتا چلتا ہے کہ کون سا آپریشن پہلے انجام دینا ہے۔ مختلف اوپریٹرز کی مختلف ترجیح ہوتی ہے۔ جن کی ترجیح زیادہ ہو انھیں پہلے حل کیا جاتا ہے اور جن کی ترجیح سب سے کم ہو انھیں آخر میں حل کیا جاتا ہے۔

# مشق

## سوال نمبر 1: کثیر الانتخابی سوالات۔

1۔ printf ________ قسم کا ڈیٹا پرنٹ کرنے کے لیے استعمال ہوتا ہے۔

الف) int　ب) float　ج) char　د) پہلے تینوں

2۔ scanf C- پروگرامنگ لینگوئج میں ________ ہے۔

الف) مطلوبہ لفظ　ب) لائبریری　ج) فنکشن　د) کوئی بھی نہیں

3۔ getch() صارف سے ______ ان پٹ لینے کے لیے استعمال ہوتا ہے۔

الف) int　ب) float　ج) char　د) پہلے تینوں

4۔ کوڈ (Code) کا یہ حصہ ایگزیکیوٹ ہونے کے بعد متغیر a کی قیمت کیا ہوگی؟

```
int a = 4;
float b = 2.2;
a = a * b;
```

الف) 8.8　ب) 8　ج) 8.0　د) 8.2

5۔ ان میں سے کوڈ کی کونسی لائن صحیح ہے؟

الف) int=20;　ب) grade='A'　ج) line=this is a line　د) کوئی بھی نہیں

6۔ ان میں سے کس آپریٹر کی ترجیح سب سے زیادہ ہے؟

الف) /　ب) =　ج) >　د) !

7۔ ان میں سے کون سی آپشن آپریٹر کی قسم نہیں ہے؟

الف) ارتھمیٹک آپریٹر　ب) ریلیشنل آپریٹر　ج) چیک آپریٹر　د) لاجیکل آپریٹر

8۔ آپریٹر % __________ کیلکولیٹ کرنے کے لیے استعمال ہوتا ہے۔

الف) پرسنٹیج　ب) ریمنڈر (بقیہ رقم)　ج) فیکٹوریل　د) مربع

9۔ ان میں سے کون سا کریکٹر C- لینگوئج میں درست ہے۔

الف) "here"　ب) "a"　ج) 'a'　د) کوئی بھی نہیں

10) C لینگوئج کے بارے میں کونسی آپشن درست ہے؟

(الف) C ایک کیس سینسٹو (case Sensitive) لینگوئج نہیں ہے۔
(ب) کی۔ ورڈز کو ویری ایبلز کے نام کے طور پر استعمال کیا جا سکتا ہے۔
(ج) تمام لاجیکل آپریٹرز بائنری آپریٹرز ہوتے ہیں۔
(د) کوئی بھی نہیں۔

## سوال نمبر 2: غلط درست کا انتخاب کریں۔

1) ایک انٹیجر کی قیمت زیادہ سے زیادہ 32000 ہو سکتی ہے۔ درست/غلط
2) فارمیٹ سپیسفائرز کے شروع میں % سائن آتا ہے۔ درست/غلط
3) تقسیم کے آپریٹر کی ترجیح ضرب کے آپریٹر سے زیادہ ہے۔ درست/غلط
4) getch() صارف سے ہر قسم کا ڈیٹا ان پُٹ لینے کے لیے استعمال ہوتا ہے۔ درست/غلط
5) scanf آؤٹ پُٹ آپریشنز کے لیے استعمال ہوتا ہے۔ درست/غلط

## سوال نمبر 3: درج ذیل کی تعریف لکھیں۔

(1) سٹیٹمنٹ ٹرمینیٹر (2) فارمیٹ سپیسفائر (3) اسکیپ سیکوئنس
(4) Scan f (5) ماڈولس آپریٹر

## سوال نمبر 4: ان سوالوں کے مختصر جواب دیں۔

1) Scanf اور getch() میں کیا فرق ہے؟
2) C- لینگوئج کا کونسا فنکشن سکرین پر آؤٹ پُٹ دکھانے کے لیے استعمال ہوتا ہے؟
3) ان پُٹ آؤٹ پُٹ آپریشنز میں فارمیٹ سپیسفائرز بتانا کیوں ضروری ہے؟
4) اسکیپ سیکوئنسز کیا ہوتے ہیں؟ ہمیں ان کی ضرورت کیوں ہوتی ہے؟
5) ارتھمیٹک آپریشنز میں کونسے آپریٹرا استعمال ہوتے ہیں؟
6) ریلیشنل آپریٹرز کیا ہیں؟ مثال دے کر وضاحت کریں۔
7) لاجیکل آپریٹرز کیا ہیں؟ مثال دے کر وضاحت کریں۔
8) یونری آپریٹرز اور بائنری آپریٹرز میں کیا فرق ہے؟
9) == آپریٹرز اور = آپریٹرز میں کیا فرق ہے؟
10) آپریٹرز کی ترجیح سے کیا مراد ہے؟ C- لینگوئج میں کس آپریٹر کی ترجیح سب سے زیادہ ہے؟

## سوال نمبر 5: کوڈ کے ان حصوں کی آؤٹ پُٹ لکھیں۔

**a)**
```
# include<stdio.h>
void main()
{
        int x= 2, y= 3, z= 6;
        int ans1,ans2 , ans3 ;
        s1 = x / z * y ;
        ans2 = y + z / y * 2;
        ans3 = z / x + x * y;
        printf(“%d %d %d”, ans1, ans2, ans3 );
}
```

**b)**
```
# include<stdio.h>
void main ()
{
        printf ( "nn \n\n nnn\nn\nt\t" ) ;
        printf ( "nn /n/n nn/n\n" ) ;
}
```

**c)**
```
#include<stdio.h>
void main()
{
        int a = 4, b;
        float c = 2.3;
        b = c * a;
        printf(“%d”, b);
}
```

**d)**
```
#include<stdio.h>
void main()
{
        int a = 4 * 3 / ( 5 + 1 ) + 7 % 4;
        printf(“%d”, a);
}
```

**e)**
```
#include<stdio.h>
void main()
{
        printf(“%d”, (( ( 5 > 3 ) && (4 > 6) ) || (7 > 3)) );
}
```

## سوال نمبر 6: کوڈ کے درج ذیل حصوں میں ایررز تلاش کریں۔

**a)**
```
#include<stdio.h>
void main ()
{
        int a , b = 13;
        b = a % 2;
        printf(“Value of b is : %d, b);
}
```

**b)**
```
#include<stdio.h>
void main ()
{
        int a , b , c,
        printf(“Enter First Number: ”);
        scanf(“%d”, &a);
        printf(“Enter second number : ”);
        scanf(“%d”, &b);
        a + b = c;
]
```

**c)**
```
#include<stdio.h>;
main ()
{
        int num;
        printf(Enter number: ”);
        scanf(%d, &num);
};
```

**d)**
```
include<stdio.h>
int main ()
{
        float f;
        printf[“Enter value: ”];
        scanf(“%c”, &f);
}
```

## پروگرامنگ کی مشقیں

**مشق 1:**

ایک کمپنی میں تنخواہ کا تعین کرنے کا طریقہ کار نیچے دیا گیا ہے۔

*Basic Salary = Pay Rate Per Hour X Working Hours Of Employee*
*Overtime Salary = Overtime Pay Rate X Overtime Hours Of Employee*
*Total Salary = Basic Salary + Overtime Salary*

ایک پروگرام لکھیں جو ملازم کے بنیادی کام کرنے کے گھنٹے اور اضافی گھنٹے بطور ان پُٹ لے، کل تنخواہ کا حساب لگائے اور اسے سکرین پر دکھائے۔

## مشق 2:

ایک پروگرام لکھیں جو سینٹی گریڈ میں درجہ حرارت کو ان پُٹ لے اور اسے فارن ہائیٹ میں تبدیل کر کے آؤٹ پُٹ دکھائے۔

درجہ حرارت کو سینٹی گریڈ سے فارن ہائیٹ میں کرنے کا فارمولا یہ ہے۔

$$F= \frac{9}{5}c+ 32$$

## مشق 3:

ایک پروگرام لکھیں جو ایک printf سٹیٹمنٹ کو استعمال کرتے ہوئے یہ آؤٹ پُٹ دکھائے۔

```
*       *       *       *
1       2       3       4
```

## مشق 4:

ایک پروگرام لکھیں جو ایک printf سٹیٹمنٹ کو استعمال کرتے ہوئے یہ آؤٹ پُٹ دکھائے

```
I am a Boy
I live in Pakistan
I am a proud Pakistani
```

## مشق 5:

کپڑے کا ایک برینڈ ہر چیز پر 15% ڈسکاؤنٹ دیتا ہے۔ ایک خاتون اس برینڈ کی پانچ قمیصیں خریدتی ہے۔ ایک پروگرام لکھیں جو ڈسکاؤنٹ کے بعد کی کل قیمت اور ڈسکاؤنٹ کی گئی رقم کا حساب لگائے۔ قمیصوں کی اصل قیمتیں یہ ہیں۔

قمیص1 =423
قمیص2 = 320
قمیص3 = 270
قمیص4 = 680
قمیص5 = 520

**نوٹ:** قمیصوں کی قیمتیں 5 ویری ایبلز میں محفوظ کریں۔

## مشق 6:

ایک پروگرام لکھیں جو دو انٹیجر ویری ایبلز کی قیمتوں کا تیسرے ویری ایبل کے استعمال کے بغیر تبادلہ (Swap) کرے۔

## مشق 7:

ایک پروگرام لکھیں جو 5 ہندسوں والا نمبر بطور ان پُٹ لے اور پہلا اور آخری ہندسہ جمع کر کے جواب پرنٹ کرے۔

## مشق 8:

ایک پروگرام لکھیں جو ماہانہ تنخواہ اور ماہانہ اخراجات جیسے بجلی کا بل، گیس کا بل، کھانے کے اخراجات صارف سے بطور ان پُٹ لے۔ پروگرام ان سب کو کیلکولیٹ کرے۔

کل ماہانہ اخراجات

کل سالانہ اخراجات

ماہانہ بچت

سالانہ بچت

ایک مہینے کی اوسط بچت

ایک مہینے کے اوسط اخراجات

## مشق 9:

ایک پروگرام لکھیں جو ایک کریکٹر اور سٹیپس (Steps) کی تعداد صارف سے ان پُٹ لیں۔
پروگرام پھر اس کریکٹر سے اتنے سٹیپس آگے جمپ کرے۔

**SAMPLE OUTPUT**

```
Enter  Character : a

      Enter Step : 2

 New  Character : C
```

## مشق 10:

ایک پروگرام لکھیں جو دائرے کا رداس (Radius) ان پُٹ لے۔ پروگرام دائرے کا رقبہ شمار کر کے سکرین پر دکھائے۔

# باب 3

# مشروط منطق

## تدریسی مقاصد: (Student Learning Outcomes)

- کنٹرول سٹیٹمنٹ کی تعریف
- سلیکشن سٹیٹمنٹ کی تعریف
- if سٹیٹمنٹ کے سٹرکچر کا علم
- if سٹیٹمنٹ کا استعمال
- if-else سٹیٹمنٹ کے ڈھانچے کا علم
- نیسٹڈ سلیکشن سٹرکچر کا استعمال

## تعارف (Introduction)

اپنی روزمرہ زندگی میں ہم اکثر کچھ کام حالات کے حساب سے کرتے ہیں مثلاً اگر میں چھے بجے اٹھا تو میں صبح کی سیر کو جاؤں گا، اگر بادل ہوئے تو میں اپنے ساتھ چھتری لے کر جاؤں گا، اگر سارہ امتحان میں پاس ہوئی تو میں اسے گھڑی دوں گا۔ یہ سارے فیصلے شرائط کی بنا پر کیے جاتے ہیں۔ اگر شرط پوری ہوئی تو میں یہ کام کروں گا ورنہ نہیں۔ بعض اوقات اگر شرط پوری نہ ہو تو ہم کوئی اور کام کرتے ہیں۔ اسے مشروط منطق کہتے ہیں۔ اس باب میں ہم سیکھیں گے کہ C- پروگرامنگ لینگوئج میں مشروط منطق کا استعمال کیسے ہوتا ہے۔

## 3.1 کنٹرول سٹیٹمنٹس (Control Statements)

ایک سوال کو حل کرنے کے لیے ہمیں پروگرام کے تسلسل کو کنٹرول کرنا پڑتا ہے۔ بعض اوقات ہم ہدایات (instructions) کا ایک سیٹ چلاتے ہیں اگر ایک مخصوص شرط پوری ہو، اور اگر وہ پوری نہ ہو تو ہدایات کا دوسرا سیٹ (مجموعہ) چلاتے ہیں۔ مزید یہ کہ بعض اوقات ہم سٹیٹمنٹس کے ایک حصے کو ایک خاص حد تک دُہراتے ہیں۔ ہم کنٹرول سٹیٹمنٹس کے ذریعے پروگرام کا تسلسل کنٹرول کر سکتے ہیں۔

C- لینگوئج میں کنٹرول سٹیٹمنٹس کی تین اقسام ہیں:

1۔ سیکوئینشل (Sequential) کنٹرول سٹیٹمنٹس
2۔ کنڈیشنل (Conditional) کنٹرول سٹیٹمنٹس
3۔ ریپیٹیشن (Repition) کنٹرول سٹیٹمنٹس

سیکوئینشل کنٹرول C- لینگوئج کا پہلے سے طے شدہ کنٹرول سٹرکچر ہے۔ سیکوئینشل کنٹرول کے مطابق تمام سٹیٹمنٹس دی گئی ترتیب کے مطابق چلتی ہیں۔ اب تک ہم نے صرف سیکوئینشل کنٹرول کے مطابق کام کیا ہے۔ اس باب میں ہم سلیکشن کنٹرول سٹیٹمنٹس پر غور کریں گے۔

## 3.2 سلیکشن سٹیٹمنٹس (Selection Statement)

وہ سٹیٹمنٹس جو شرائط کی بنا پر ہماری فیصلہ کرنے میں مدد کرتی ہیں کہ آگے کونسی سٹیٹمنٹس چلنی چاہییں سلیکشن سٹیٹمنٹس کہلاتی ہیں۔

مشروط سٹیٹمنٹس کی دو قسمیں یہ ہیں:

1۔ If سٹیٹمنٹ
2۔ If-else سٹیٹمنٹ

## 3.2.1 If سٹیٹمنٹ:

C-لینگویج میں ہم If سٹیٹمنٹ کے ذریعے شرط بتا کر اس سے کوڈ منسوب کر سکتے ہیں۔ یہ کوڈ تب چلتا ہے اگر شرط پوری ہو جائے ورنہ نہیں چلتا۔

### if سٹیٹمنٹ کا ڈھانچہ (Structure):

C-لینگویج میں if سٹیٹمنٹ کا ڈھانچہ یہ ہے۔

```
if (شرط)
    منسوب کیا گیا کوڈ
```

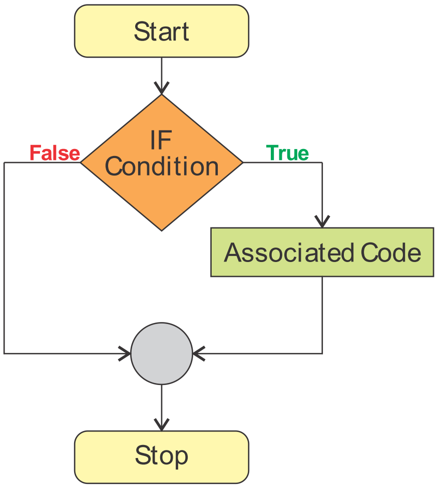
If سٹیٹمنٹ ڈھانچے کے مختلف حصوں کی مختصر تفصیل یہ ہے۔

1) دیے گئے سٹرکچر میں If ایک کی-ورڈ ہے جس کے بعد واوین میں شرط لکھی جاتی ہے۔
2) ایک شرط کوئی بھی درست ایکسپریشن ہو سکتی ہے جیسے ارتھمیٹک ایکسپریشنز، ریلیشنل ایکسپریشنز، منطقی ایکسپریشنز یا ان سب کا مجموعہ درست ایکسپریشنز کی کچھ مثالیں یہاں دی گئی ہیں۔ جنھیں ایک شرط کے طور پر استعمال کیا جا سکتا ہے۔

| | | |
|---|---|---|
| **a-** | 5 | (true) |
| **b-** | 5 + 4 | (true) |
| **c-** | 5 – 5 | (false) |
| **d-** | 5 > 4 | (true) |
| **e-** | 5 == 4 | (false) |
| **f-** | ! (4 > 5) | (true) |
| **g-** | (5 > 4) && (10 < 9) | (false) |
| **h-** | (5 > 4) \|\| (9 < 10) | (true) |

ایک ایکسپریشن جس کی قیمت صفر نہ ہو وہ true ہوتا ہے جیسے اوپر ایکسپریشن a اور b True ہیں جب کہ ایکسپریشن c کا نتیجہ false ہے۔ ایکسپریشن میں ویری ایبلز بھی استعمال کیے جا سکتے ہیں۔ اس صورت میں ویری ایبلز کی قیمتوں کو استعمال کرتے ہوئے ایکسپریشن کی True/false قیمت نکالی جاتی ہے۔

3) منسوب کردہ کوڈ C- لینگویج کی درست سٹیٹمنٹس کا ایک مجموعہ ہوتا ہے اس میں ایک یا ایک سے زیادہ سٹیٹمنٹس ہوسکتی ہیں۔ اس فلو چارٹ میں if سٹیٹمنٹ کا بنیادی تسلسل دکھایا گیا ہے۔



اگر ہم ایک if سٹیٹمنٹ سے ایک سے زیادہ سٹیٹمنٹس منسوب کرنا چاہتے ہیں تو انھیں بلاک کے اندر لکھتے ہیں۔ لیکن اگر صرف ایک ہی سٹیٹمنٹ منسوب کرنی ہو تو اسے ہم بلاک میں لکھ تو سکتے ہیں لیکن یہ ضروری نہیں ہوتا یہ اس مثال میں واضح کیا گیا ہے۔

**EXAMPLE CODE 3.1 <\>**

```
#include<stdio.h>
void main()
{
      int a = 12;
      if (a % 2 == 0)
      {
            printf(“The variable a contains an even value.”);
            printf(“\nYou are doing a great job.”);
      }
}
Output:
The variable a contains an even value.
You are doing a great job.
```

کیونکہ جب ہم 12 کو 2 پہ تقسیم کرتے ہیں تو 0 بچتا ہے اس لیے If کے بعد دی گئی شرط پوری ہو جاتی ہے۔ چونکہ دونوں printf سٹیٹمنٹس { } بلاک کے اندر ہیں اس لیے دونوں سٹیٹمنٹس چلیں۔ اب اس کوڈ کو دیکھیں۔

**EXAMPLE CODE 3.2**

```
#include<stdio.h>
void main()
{
        int a = 4;
        int b = 5;
        if (a > b)
                printf("The value of a is greater than b.");
        printf("\nYou are doing a great job.");
}
```

**Output:**

```
You are doing a great job.
```

جیسا کہ شرط پوری نہیں ہوئی اور سٹیٹمنٹس جو If سٹیٹمنٹ کے بعد آ رہی ہیں وہ { } بلاک کے اندر نہیں اس لیے صرف { } بلاک کے بغیر والی دوسری سٹیٹمنٹ چلی کیونکہ پروگرام نے صرف پہلی سٹیٹمنٹ کو if سٹیٹمنٹ سے منسوب کیا۔

### C-میں if سٹیٹمنٹ کا استعمال:

ہم if سٹیٹمنٹ کے تصور کو سمجھنے کے لیے مختلف مثالوں کو دیکھتے ہیں۔

**3.1 پروگرامنگ ٹائم (Programming Time)**

**سوال:** ایک پروگرام لکھیں جو طالب علم کی پرسنٹیج اِن پُٹ لے اور اگر پرسنٹیج 50 سے زیادہ ہو تو "Pass" پرنٹ کرے۔

```
#include <stdio.h>
void main()
{
        float percentage;
        printf ("Enter the percentage: ");
```

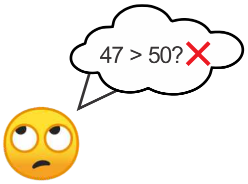
جاری ہے۔

```
        scanf ("%f", &percentage);
        if (percentage > 50)
              printf ("PASS\n");
  }
```

### آؤٹ پُٹ:

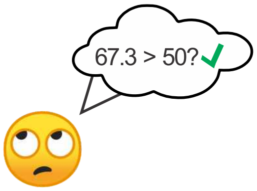
جب 47 اِن پُٹ دی گئی تو پروگرام ختم ہوگیا کیونکہ 47، 50 سے کم ہے اور شرط پوری نہیں ہوئی۔

```
Enter the percentage : 47
```



جب 67.3 اِن پُٹ دی گئی تو کنسول پر "Pass" پرنٹ ہوا کیونکہ 67.3، 50 سے بڑا ہے اور شرط پوری ہوگئی۔

```
Enter the percentage : 67.3
PASS
```



### 3.2 پروگرامنگ ٹائم (Programming Time)

ایک مارکیٹنگ کرنے والی کمپنی اپنے ملازمین کی تنخواہ مندرجہ ذیل فارمولے کی مدد سے بناتی ہے۔

Gross Salary = Basic Salary+( Number of Items Sold x 8 )+ Bonus

اگر بیچی جانے والی اشیا کی تعداد 100 سے زیادہ ہے اور ٹوٹنے والی اشیا کی تعداد 0 ہے تو بونس (Bonus) 10، 000 ہوگا نہیں تو بونس 0 ہوگا۔

ایک پروگرام لکھیں جو کہ Basic Salary، بیچی جانے والی اور ٹوٹنے والی اشیا (No. of Solid Items, No.of Broken Items) بطور اِن پٹ صارف سے لے اور ملازم کی Gross Salary کا شمار کرے اور پرنٹ کرے۔

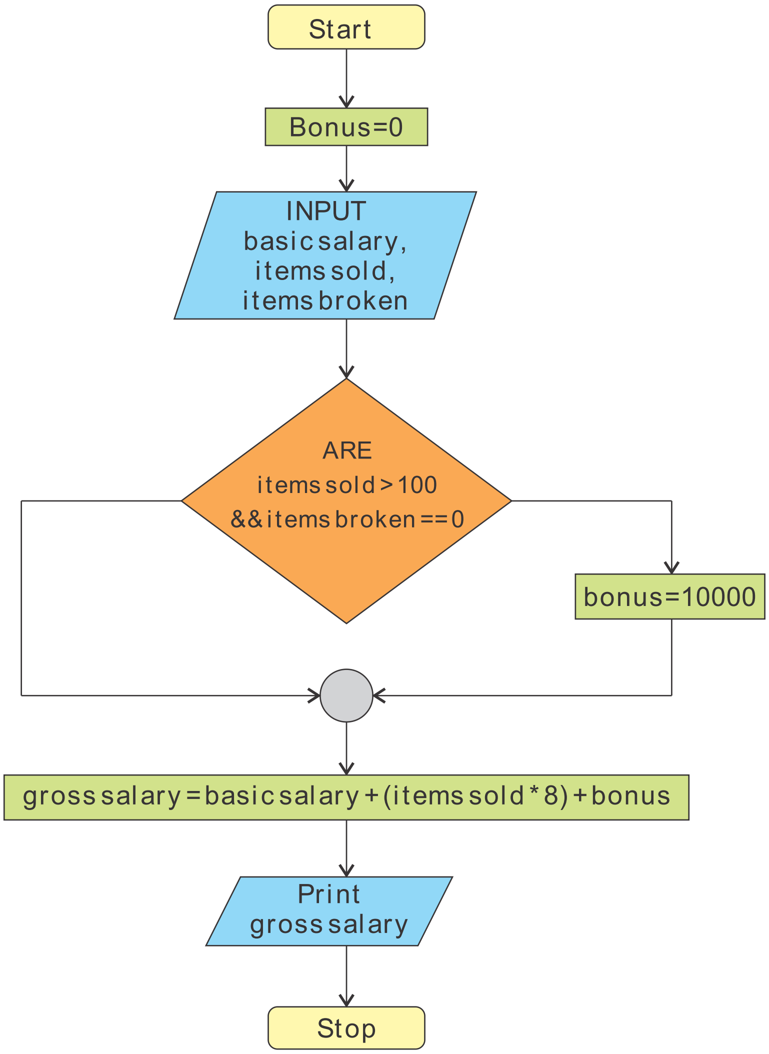
جاری ہے۔

```
#include<stdio.h>
void main()
{
        int  basic_salary,  items_sold,  items_broken,
        gross_salary;
        int bonus = 0;
        printf(“Enter the basic salary: “);
        scanf(“%d”, &basic_salary);
        printf(“Enter the number of items sold: “);
        scanf(“%d”, &items_sold);
        printf(“Enter the number of items broken: “);
        scanf(“%d”, &items_broken);
        if (items_sold > 100 && items_broken == 0)
                bonus = 10000;
        gross_salary = basic_salary + (items_sold * 8) + bonus;
        printf(“Gross  salary  of  the  employee  is  %d”,
        gross_salary);
}
```

**وضاحت:**

اوپر دی گئی مثال میں بونس کو 0 سے انشلائز کیا گیا ہے۔ کیونکہ اگر بیچی جانے والی اشیا کی تعداد 100 سے کم ہو تو ی 0 تصوّر ہوگا۔ if سٹیٹمنٹ کے اندر یہ دیکھا جاتا ہے کہ کیا بیچی جانے والی اشیا کی تعداد 100 سے زیادہ ہے اور اگر ایسا ہی ہو تو بونس (Bonus) 10،000 مقرر ہوتا ہے۔ Gross Salary کا حساب کتاب if بلاک { } باہر کیا گیا ہے۔ یہ اس لیے ہے کہ Gross Salary کا حساب کتاب ہونا ہی ہے چاہے، بیچی جانے والی اشیا 100 سے زیادہ ہوں یا کم۔

مندرجہ ذیل تصویر پروگرام کے فلو چارٹ کو ظاہر کر رہی ہے۔


Start
Bonus=0
INPUT
basic salary,
items sold,
items broken
ARE
items sold > 100
&& items broken == 0
bonus=10000
gross salary = basic salary + (items sold * 8) + bonus
Print
gross salary
Stop

**سرگرمی 3.1:**

ایسا پروگرام لکھیں جو کسی شخص کی عمر اِن پٹ کے طور پر لے اور سکرین پر "Teenager" ظاہر کرے اگر عمر 13 سے 19 سال کے درمیان ہو۔

**سرگرمی 3.2:**

ایک پروگرام لکھیں جو سال کو اِن پٹ کے طور پر لے۔ اگر یہ سال لِیپ کا ہو تو سکرین پر "Leap Year" ظاہر کرے۔ لِیپ کا سال وہ سال ہوتا جو 4 پر مکمل تقسیم ہو جائے۔

**اہم نکتہ**

If سٹیٹمنٹ کے اندر والی سٹیٹمنٹ کو ٹیب کے ذریعے صحیح طرح سے انڈینٹ (indent) کریں۔ اس سے پروگرام کو سمجھنا آسان ہو جاتا ہے۔

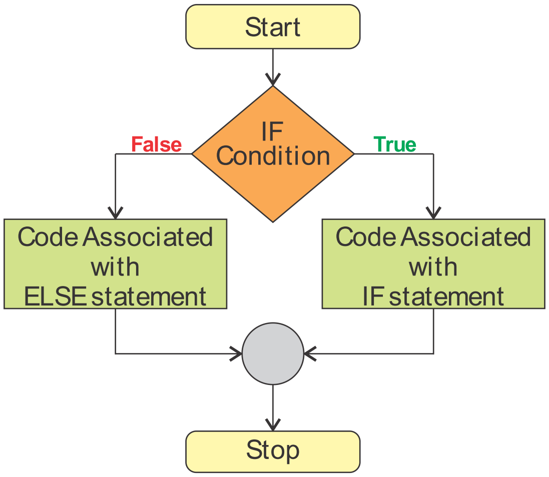
### 3.2.2 if-else سٹیٹمنٹ:

اب تک ہم نے دیکھا کہ کس طرح سے ایک شرط پوری ہونے پر ہدایات کا ایک سیٹ چلایا جاتا ہے لیکن اگر شرط پوری نہ ہو تو ہم کچھ نہیں کرتے۔ اگر ہم چاہتے ہوں کہ ہدایات کا ایک سیٹ تب چلے اگر شرط پوری ہو جائے ورنہ کوئی دوسرا سیٹ چلے تو ایسی صورت میں ہم If-else سٹیٹمنٹ استعمال کرتے ہیں۔ اس میں اگر شرط پوری ہو جائے تو If سٹیٹمنٹ سے منسوب کردہ سٹیٹمنٹس چلتی ہیں ورنہ else سٹیٹمنٹ سے منسوب کردہ سٹیٹمنٹس چلتی ہیں۔

If-else سٹیٹمنٹ کا ڈھانچہ یہ ہے۔

(کنڈیشن) If
منسوب کردہ کوڈ
else
منسوب کردہ کوڈ

اگر شرط پوری ہو تو if سٹیٹمنٹ سے منسوب کردہ کوڈ چلے گا ورنہ else سٹیٹمنٹ سے منسوب کردہ کوڈ چلے گا۔ اس فلو چارٹ سے If-else سٹیٹمنٹ کا ڈھانچہ ظاہر ہوتا ہے۔


Start
IF Condition
False
True
Code Associated with ELSE statement
Code Associated with IF statement
Stop

**اہم نوٹ:**

ممکن ہے کہ ایک If سٹیٹمنٹ کے ساتھ else سٹیٹمنٹ نہ ہو مگر else سٹیٹمنٹ کے لیے If سٹیٹمنٹ ہونا ضروری ہے۔

else کی-ورڈز سے پہلے اگر If کے اندر ایک سے زیادہ سٹیٹمنٹس آرہی ہوں تو انہیں {} بلاک کے اندر لکھنا ضروری ہوتا ہے ورنہ کمپائلر ایرر دیتا ہے۔ اس تصور کو سمجھنے کے لیے نیچے دی گئی مثال پر غور کریں۔

**EXAMPLE CODE 3.3**

```
#include<stdio.h>
void main()
{
        int a = 15;

        if (a % 2 == 0)
                printf(“The variable a contains an even value.”);
                printf(“\nYou are doing a great job.”);
        else
                printf(“The variable a contains an odd value.”);
}
```

**ERROR** else without an associated if

اوپر دیا گیا کوڈ کمپائل نہیں کیا جا سکتا کیونکہ جیسا کہ پہلے ذکر کیا گیا کہ {} بلاک کے بغیر If سٹیٹمنٹ کے ساتھ صرف ایک ہی سٹیٹمنٹ منسوب کی جا سکتی ہے۔ اس صورت میں پہلی سٹیٹمنٹ یعنی ("Printf(" VariableThe If سٹیٹمنٹ سے منسوب ہے لیکن دوسری سٹیٹمنٹ یعنی ("Printf("\nyou are not doing great jobThe سٹیٹمنٹ سے منسوب نہیں ہے۔ اس لیے else والا حصہ بھی if سٹیٹمنٹ سے منسلک نہیں ہے۔ ہم جانتے ہیں کہ else بلاک کا if سٹیٹمنٹ سے منسلک ہونا ضروری ہے۔ اس مسئلے کو حل کرنے کے لیے ہم کی-ورڈ else سے پہلے والی دونوں سٹیٹمنٹ کو {} بلاک کے اندر رکھ سکتے ہیں۔

**کیا آپ جانتے تھے؟**

قوسین کے اندر لکھا گیا ایک سے زیادہ ہدایات کا مجموعہ بلاک یا کمپاؤنڈ سٹیٹمنٹ کہلاتا ہے۔

درج ذیل کوڈ اس کی وضاحت کرتا ہے۔

**EXAMPLE CODE 3.4**

```
#include<stdio.h>
void main()
{
    int a = 15;
    if (a % 2 == 0)
    {
        printf(“The variable a contains an even value.”);
        printf(“\nYou are doing a great job.”);
    }
    else
        printf(“The variable a contains an odd value.”);
}
```

**Output:**

```
The variable a contains an odd value.
```

## If-else سٹیٹمنٹ کا استعمال:

نیچے دیا گیا پروگرام ایک کریکٹر اِن پٹ لیتا ہے اور اگر وہ ہندسہ ہو تو "DIGIT" پرنٹ کرتا ہے ورنہ "NOT DIGIT" پرنٹ کرتا ہے۔

**EXAMPLE CODE 3.5**

```
#include <stdio.h>
void main()
{
    char input;
    printf (“Please enter a character: “);
    scanf (“%c”, &input);
    if (input >= '0' && input <= '9')
        printf (“DIGIT\n”);
    else
        printf (“NOT DIGIT\n”);
}
```

جاری ہے۔

اگر صارف ایک ہندسہ درج کرے جیسا کہ 5 تو سکرین پر "DIGIT" ظاہر ہوگا۔

اگر صارف کوئی اور کریکٹر درج کرے جیسا کہ k تو "NOT DIGIT" ظاہر ہوگا کیونکہ شرط پوری نہیں ہوئی۔

```
Please enter a character: k
        NOT DIGIT
```

**سرگرمی 3.3:**

ایک پروگرام لکھیں جو ایک شخص کے جسم کا درجہ حرارت اِن پُٹ لے اور اگر اِن پُٹ 98.6 سے زیادہ ہو تو "You have fever" پرنٹ کرے ورنہ "You don't have fever" پرنٹ کرے۔

**اہم نکتہ**

اگر if سٹیٹمنٹ یا else سٹیٹمنٹ کے اندر ایک سے زیادہ ہدایات ہوں تو انھیں ایک بلاک کی صورت میں لکھا جاتا ہے ورنہ کمپائلر صرف ایک ہدایت کو اس سے منسوب کرتا ہے اور باقی ہدایات کو ان سے الگ سمجھتا ہے۔

**اہم نوٹ:**

C- لینگوئج میں ایک If-else-if سٹیٹمنٹ بھی ہوتی ہے جس کا ڈھانچہ یہ ہوتا ہے۔

if (پہلی کنڈیشن)

یہ کوڈ جو تب چلے گا اگر پہلی شرط پوری ہو۔

else if (دوسری کنڈیشن)

یہ کوڈ جو تب چلے گا اگر پہلی شرط نہ پوری ہو لیکن دوسری شرط پوری ہو۔

-----------------

else if (Nth کنڈیشن)

یہ کوڈ جو تب چلے گا اگر پچھلی شرطیں پوری نہ ہوں بس Nth شرط پوری ہو۔

else

یہ کوڈ جو تب چلے گا اگر کوئی شرط پوری نہ ہو۔

## 3.3 پروگرامنگ ٹائم (Programming Time)

ایک پروگرام لکھیں جو طالب علم کے نمبروں کی فیصد اِن پُٹ لے اور دیے گئے ٹیبل کے مطابق گریڈ بتائے۔

| Percentage | Grade |
|---|---|
| 80% and above | A |
| 70% - 80% | B |
| 60% - 70% | C |
| 50% - 60% | D |
| Below 50% | F |

**Program:**

```
#include<stdio.h>
void main()
{
      float percentage;
      printf (“Enter the percentage: ”);
      scanf (“%f”, &percentage);
      if (percentage >= 80)
            printf (“A\n”);
      else if (percentage >= 70)
            printf (“B\n”);
      else if (percentage >= 60)
            printf (“C\n”);
      else if (percentage >= 50)
            printf (“D\n”);
      else
            printf (“F\n”);
}
```

### 3.2.3 نیسٹڈ سلیکشن سٹرکچرز (Nested Selection Structures)

ہم If-else سٹیٹمنٹ کے دیے گئے ڈھانچے کو غور سے دیکھتے ہیں۔

```
If (کنڈیشن)
    منسوب کردہ کوڈ
else
    منسوب کردہ کوڈ
```

if یا else سٹیٹمنٹ سے منسوب کردہ کوڈ میں سٹیٹمنٹس کا کوئی بھی سیٹ آ سکتا ہے جو C- لینگوئج کے اعتبار سے درست ہو۔ اس کا مطلب ہے کہ if یا else بلاک کے اندر ہم مزید if یا if-else سٹیٹمنٹس لکھ سکتے ہیں۔ اس کا مطلب یہ بھی ہے کہ ہم ان اندر والی if یا if-else سٹیٹمنٹس کے اندر بھی مزید if یا if-else سٹیٹمنٹس لکھ سکتے ہیں اور یہ کام ہم جتنی بار چاہیں دُہرا سکتے ہیں۔ کنڈیشنل سٹیٹمنٹس کے اندر کنڈیشنل سٹیٹمنٹس کو ہم نیسٹڈ سلیکشن سٹرکچرز کہتے ہیں۔ نیچے دیے گئے تمام ڈھانچے نیسٹڈ سلیکشن سٹرکچر کے منسوب کردہ کوڈ ڈھانچے ہیں۔

```
if (پہلی کنڈیشن)
    if (دوسری کنڈیشن)
        منسوب کردہ کوڈ
    else
    if (تیسری کنڈیشن)
        منسوب کردہ کوڈ
else
    منسوب کردہ کوڈ
```

```
if (پہلی کنڈیشن)
    if (دوسری کنڈیشن)
        منسوب کردہ کوڈ
else
    منسوب کردہ کوڈ
```

```
if (پہلی کنڈیشن)
    if (دوسری کنڈیشن)
        منسوب کردہ کوڈ
    else
        منسوب کردہ کوڈ
else
    if (تیسری کنڈیشن)
        منسوب کردہ کوڈ
    else
        منسوب کردہ کوڈ
```

```
if (پہلی کنڈیشن)
    if (دوسری کنڈیشن)
        منسوب کردہ کوڈ
    else
        منسوب کردہ کوڈ
else
    if (تیسری کنڈیشن)
        منسوب کردہ کوڈ
```

## نیسٹڈ سلیکشن سٹرکچرز کا استعمال

نیسٹڈ سلیکشن سٹرکچرز کا استعمال سمجھنے کے لیے ہم ان مثالوں پر نظر ڈالتے ہیں۔

### 3.4 پروگرامنگ ٹائم (Programming Time)

**پرابلم:** بجلی کے بل کی کمپنی اس فارمولے کے مطابق بجلی کے بل کا حساب لگاتی ہے۔

Bill Amount = No. of Unit Consumed x Unit Price

بل کی رقم: استعمال شدہ یونٹس کی تعداد x ایک یونٹ کی قیمت

بجلی کے صارفین کی دو قسمیں ہیں یعنی کمرشل اور گھریلو صارفین۔ گھریلو صارفین کے لیے ایک یونٹ کی قیمت اس حساب سے رکھی جاتی ہے۔

| Units Consumed | Unit Price |
|---|---|
| Units < = 200 | Rs 12 |
| Units > 200 but Units < = 400 | Rs 15 |
| Units > 400 | Rs 20 |

کمرشل صارفین کے لیے ایک یونٹ کی قیمت اس حساب سے رکھی جاتی ہے۔

| Units Consumed | Unit Price |
|---|---|
| Units < = 200 | Rs 15 |
| Units > 200 but Units < = 400 | Rs 20 |
| Units > 400 | Rs 24 |

ایک پروگرام لکھیں جو صارف کی قسم اور استعمال شدہ یونٹس کی تعداد ان پٹ لے اور صارف کا بجلی کا بل بتائے۔

**Program:**

```
#include<stdio.h>
void main()
{
	int units, unit_price, bill;
	char user_type;
	printf(“Please enter h for home user and c for commercial
	user: ”);
	scanf(“%c”, &user_type);
	printf(“Please enter the number of units consumed: “);
	scanf(“%d”, &units);
```

جاری ہے۔

```
      if(units <= 200)
            if(user_type == 'h')
                  unit_price = 12;
            else if(user_type == 'c')
                  unit_price = 15;
      else if(units > 200 && units <= 400)
            if(user_type == 'h')
                  unit_price = 15;
            else if(user_type == 'c')
                  unit_price = 20;
      else
            if(user_type == 'h')
                  unit_price = 15;
            else if(user_type == 'c')
                  unit_price = 24;
      bill = units * unit_price;
      printf(“Your electricity bill is %d”, bill);
}
```

**اہم نکتہ**

کمپاؤنڈ سٹیٹمنٹس میں ایک عام غلطی یہ ہے کہ ٹائپنگ کے دوران ایک یا دونوں قوسین لگانا بھول جانا۔ اس ایرر سے بچنے کے لیے بہتر ہے کہ پہلے قوسین ٹائپ کریں اور پھر بلاک میں سٹیٹمنٹس لکھیں۔

**سرگرمی 3.4:**

ایک یونیورسٹی کے انڈر گریجویٹ پروگرام کے لیے اہلیت کا معیار یہ ہے۔

**BSSE پروگرام:** انٹرمیڈیٹ میں 80% یا اس سے زیادہ نمبر

**BSCS پروگرام:** انٹرمیڈیٹ میں 70% یا اس سے زیادہ نمبر

**BSIT پروگرام:** انٹرمیڈیٹ میں 60% یا اس سے زیادہ نمبر

ورنہ طالب علم یونیورسٹی کے کسی بھی پروگرام میں داخلہ نہیں لے سکتا۔

ایک پروگرام لکھیں جو انٹرمیڈیٹ کے نمبروں کی پرسنٹیج لے اور بتائے کہ طالب علم کن کن پروگرامز میں درخواست دینے کی اہلیت رکھتا ہے۔

**کیا آپ جانتے تھے؟**

C-پروگرامنگ لینگوئج میں کنڈیشنز کے استعمال کے لیے سوئچ کیس سٹرکچر (Switch case structure) بھی ہوتا ہے لیکن یہ صرف محدود منظر ناموں کے لیے استعمال ہو سکتا ہے جبکہ if اور if-else سٹرکچرز فیصلوں پر مبنی تمام منظر ناموں پر استعمال ہو سکتے ہیں۔

**سرگرمی 3.5:**

ایک پروگرام لکھیں جو دو انٹیجر ان پٹ لے اور صارف سے کہے کہ 1 سے 4 کے درمیان اپنا انتخاب درج کرے۔ پھر پروگرام دیے گئے ٹیبل کے مطابق اوپریشن انجام دے اور نتیجہ بتائے۔

| Choice | Operation |
|---|---|
| 1 | Addition |
| 2 | Subtraction |
| 3 | Multiplication |
| 4 | Division |

## 3.2.4 حل شدہ مثالیں

**3.5 پروگرامنگ ٹائم (Programming Time)**

**پرابلم:** ایک پروگرام لکھیں جو بتائے کہ تین نمبروں میں سے سب سے بڑا نمبر کونسا ہے۔

**Program:**

```
include <stdio.h>
void main()
{
      int n1, n2, n3;
      printf ("Enter three numbers");
      scanf ("%d%d%d", &n1, &n2, &n3);
      if (n1 > n2 && n1 > n3)
            printf ("The largest number is %d", n1);
      else if (n2 > n3 && n2 > n1)
            printf ("The largest number is %d", n2);
      else
            printf ("The largest number is %d", n3);
}
```

## 3.6 پروگرامنگ ٹائم (Programming Time)

**پرابلم:** ایک پروگرام لکھیں جو صارف کے انتخاب کے مطابق کیوب، سلنڈر یا دائرے کے حجم کا شمار کرے۔

**Program:**

```
#include<stdio.h>
void main()
{
        int choice;
        float volume;
        printf ("Find Volume\n");
        printf  ("1.Cube\n2.Cylinder\n3.Sphere\nEnter  your  choice:
        ");
        scanf ("%d", &choice);
        if (choice == 1)
        {
                float length;
                printf ("Enter Length: ");
                scanf ("%f", &length);
                volume = length * length * length;
                printf ("Volume is %f", volume);
        }
        else if (choice == 2)
        {
                float length1, radius1;
                printf ("Enter Length: ");
                scanf ("%f", &length1);
                printf ("Enter Radius: ");
                scanf ("%f", &radius1);
                volume = 3.142 * radius1 * radius1 * length1;
                printf ("Volume is %f", volume);
        }
```

جاری ہے۔

```
        else if (choice == 3)
        {
            float radius;
            printf ("Enter Radius: ");
            scanf ("%f", &radius);
            volume = 3.142 * radius * radius * radius;
            printf ("Volume is %f", volume);
        }
        else
            printf (“Invalid Choice”);
    }
```

**سرگرمی 3.6:**

ایک پروگرام لکھیں جو صارف کے انتخاب کے مطابق مثلث، متوازی الاضلاع، معیّن یا ٹریپیزیم کا رقبہ معلوم کرے اور سکرین پر دکھائے۔

## خلاصہ:

- پروگرام کا تسلسل کنٹرول سٹیٹمنٹس کے ذریعے کنٹرول کیا جاتا ہے۔
- **سیکوئینشل کنٹرول** C- لینگوئج کا پہلے سے طے شدہ سٹرکچر ہے۔ اس کے مطابق تمام سٹیٹمنٹس دی گئی ترتیب کے مطابق ایگزیکیوٹ ہوتی ہیں۔
- وہ سٹیٹمنٹس جو کنڈیشنز کی بنا پر ہمیں یہ فیصلہ کرنے میں مدد دیتی ہیں کہ آگے کون سی سٹیٹمنٹس چلنی چاہئیں کنڈیشنل سٹیٹمنٹس کہلاتی ہیں۔
- **if سٹیٹمنٹ** میں ہم ایک کنڈیشن بتاتے ہیں اور اس سے ایک کوڈ کو منسوب کر دیتے ہیں۔ یہ کوڈ تب ایگزیکیوٹ ہوتا ہے اگر یہ کنڈیشن پوری ہو جائے ورنہ یہ کوڈ نہیں ایگزیکیوٹ ہوتا۔
- ایک کنڈیشن کوئی بھی درست ایکسپریشن ہو سکتی ہے جیسے ارتھمیٹک ایکسپریشنز، ریلیشنل ایکسپریشنز، لاجیکل ایکسپریشنز یا ان سب کا مجموعہ۔
- if سٹیٹمنٹ کے منسوب کردہ کوڈ میں C- لینگوئج کی درست سٹیٹمنٹس کا سیٹ آ سکتا ہے۔
- **if-else سٹیٹمنٹ** if سٹیٹمنٹ سے منسوب کردہ کوڈ چلاتی ہے اگر کنڈیشن پوری ہو ورنہ else سٹیٹمنٹ سے منسوب کردہ کوڈ چلاتی ہے۔
- ممکن ہے کہ ایک if سٹیٹمنٹ کے ساتھ کوئی else سٹیٹمنٹ منسلک نہ ہو لیکن else سٹیٹمنٹ کے ساتھ if سٹیٹمنٹ کا ہونا ضروری ہے۔
- کنڈیشنل سٹیٹمنٹس کے اندر کنڈیشنل سٹیٹمنٹس **نیسٹڈ سلیکشن سٹرکچرز** کہلاتے ہیں۔

## مشق

### سوال نمبر 1: کثیر الانتخابی سوالات:

1) کنڈیشنل لاجک ۔۔۔۔۔۔۔۔۔۔ میں مدد دیتی ہے۔

a) فیصلوں  b) تکراروں  c) ٹریورسنگ (گزرنا)  d) پہلے تینوں

2) ۔۔۔۔۔۔۔۔۔۔۔۔ سٹیٹمنٹس بتاتی ہیں کہ پروگرام کی سٹیٹمنٹس کس ترتیب سے ایگزیکیوٹ ہوں گی۔

a) لوپ  b) مشروط  c) کنٹرول  d) پہلے تینوں

3) if سٹیٹمنٹ میں اگر کنڈیشن پوری نہ ہو تو کیا ہوتا ہے؟

a) پروگرام رُک جاتا ہے  b) انڈیکس آؤٹ آف باؤنڈ ایرر آتا ہے

c) باقی کوڈ چلنے لگتا ہے  d) کمپائلر کنڈیشن بدلنے کا مطالبہ کرتا ہے

4) ان میں سے کونسی سٹیٹمنٹ چلے گی؟

```
int a = 5;
if (a < 10)
        a++;
else
        if (a > 4)
                a--;
```

a) ;++a  b) ;--a  c) پہلی دونوں  d) کوئی نہیں

5) ان میں سے کونسی کنڈیشن یہ بتاتی ہے کہ c,a کا فیکٹر ہے یا نہیں؟

a) a%c==0  b) c%a==0  c) a*c==0  d) a+c==0

6) ایک کنڈیشن کوئی بھی ۔۔۔۔۔۔۔۔۔۔۔۔ ایکسپریشن ہو سکتی ہے۔

a) ارتھمیٹک  b) ریلیشنل  c) لاجیکل  d) ارتھمیٹک، ریلیشنل یا لاجیکل

7) اگر if سٹیٹمنٹ کے اندر ایک اور if سٹیٹمنٹ ہو تو یہ سٹرکچر ۔۔۔۔۔۔۔۔۔۔۔۔۔۔ کہلاتا ہے۔

a) نیسٹڈ  b) بوکسڈ  c) ریپیٹڈ  d) ڈیکمپوزڈ

8) قوسین میں بند ایک سے زیادہ ہدایات کا سیٹ ۔۔۔۔۔۔۔۔۔۔۔۔۔۔۔۔ کہلاتا ہے۔

a) بوکس  b) لسٹ  c) بلاک  d) جوب

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

1) کنٹرول سٹیٹمنٹس  2) کنڈیشنل سٹیٹمنٹس  3) سیکوئینشل کنٹرول

4) کنڈیشن  5) نیسٹڈ سلیکشن سٹرکچر

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

1) ہمیں کنڈیشنل سٹیٹمنٹس کی ضرورت کیوں ہوتی ہے؟

2) سیکوئینشل اور کنڈیشنل سٹیٹمنٹس میں فرق کریں۔

3) if سٹیٹمنٹ اور if-else سٹیٹمنٹ میں مثالوں کے ساتھ فرق کریں۔

4) نیسٹڈ سلیکشن سٹرکچر کا کیا استعمال ہے؟

5) if سٹیٹمنٹ کا ڈھانچہ تفصیل سے لکھیں۔

**سوال نمبر 4: کوڈ کے ان حصوں میں ایررز تلاش کریں۔ فرض کریں کہ تمام ویری ایبلز پہلے سے ڈکلیئر کیے ہوئے ہیں۔**

```
a) if (x ≥ 10)
          printf (“Good”);

b) if (a < b && b < c);
          sum = a + b + c;
   else
          multiply = a * b * c;

c) if (a < 7 < b)
          printf (“7”);

d) if (a == b &| x == y)
          flag = true;
   else
          flag = false;

e) if (sum == 60 || product == 175)
          printf (“Accepted %c), sum);
   else
          if (sum >= 45 || product > 100)
                 printf (“Considered %d” + sum);
          else
                 printf (“Rejected”);
```

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

**a)**

```
int a = 7, b = 10;
a = a + b;
if ( a > 20 && b < 20 )
      b = a + b;
printf ( “ a = %d, b = %d”, a, b);
```

**b)**

```
int x = 45;
if (x + 20 * 7 == 455)
      printf (“Look's Good”);
else
      printf (“Hope for the Best”);
```

**c)**

```
char c1 = 'Y', c2 = 'N';
int n1 = 5, n2 = 9;
n1 = n1 + 1;
c1 = c2;
if (n1 == n2 && c1 == c2)
      printf (“%d = %d and %c = %c”, n1, n2, c1, c1);
else
      if (n1 < n2 && c1 == c2)
            printf (“%d < %d and %c = %c”, n1, n2, c1, c2);
      else
            printf (“Better Luck Next Time!”);
```

**d)**

```
int a = 34, b = 32, c = 7, d = 15;
a = b + c + d;
if ( a < 100 )
    a = a * 2;
b = b * c;
c = c + d;
if ( a > b && c == d )
{
    c = d;
    b = c;
    a = b;
}
else
    if (a > b && c > d || b >= d + c)
    {
        d = c * c;
        a = b * b;
    }
printf (“a=%d, b=%d, c=%d, d=%d”, a, b, c, d);
```

**e)**

```
int x = 5, y = 7, z = 9;
if ( x % 2 == 0)
    x++;
else
    x = y + z;
printf (“ x = %d\n”, x);
if ( x % 2 == 1 && y % 2 == 1 && z % 2 == 1)
    printf (“All are Odd”);
if ( x > y || x < z )
{
    if ( x > y )
        y++;
    else
        if ( x < z )
            x++;
}
printf (“x = %d, y = %d, z = %d”, x, y, z);
```

# پروگرامنگ کی مشقیں

**مشق1:**

ایک پروگرام لکھیں جو دو انٹیجر (int) ان پٹ لے اور بتائے کہ پہلا نمبر دوسرے نمبر کا فیکٹر ہے یا نہیں۔

**مشق2:**

ایک پروگرام لکھیں جو ایک نمبر ان پٹ لے اور اگر وہ نمبر 3 کا حاصل ضرب ہو اور اس کا اکائی مقام (unit place) پر 5 ہو تو "Yes" پرنٹ کرے۔

**مشق3:**

گروسری (grocery) مارٹ پہ موجود ڈسکاؤنٹس کی فہرست یہ ہے۔

| Total Bill | Discount |
|---|---|
| 1000 | 10% |
| 2500 | 20% |
| 5000 | 35% |
| 10000 | 50% |

ایک پروگرام لکھیں جو کل بل ان پٹ لے اور بتائے کہ صارف کو کتنا ڈسکاؤنٹ ہوا اور ڈسکاؤنٹ کے بعد قیمت کیا ہے۔

**مشق4:**

ایک پروگرام لکھیں جو پروڈکٹ کی اصل قیمت اور قیمتِ فروخت ان پٹ لے اور بتائے کہ پروڈکٹ کو فائدے میں بیچا گیا یا نقصان میں۔ پروگرام یہ بھی بتائے کہ کتنے فیصد نفع یا نقصان ہوا۔

**مشق5:**

ایک پروگرام لکھیں جو مثلث کی 3 اطراف ان پٹ لے اور بتائے کہ یہ قائمۃ الزّاویہ مثلث ہے یا نہیں۔ قائمۃ الزّاویہ مثلث وہ ہے جس میں

$$(\text{وتر})^2 = (\text{بنیاد})^2 + (\text{اونچائی})^2$$

**مشق6:**

ایک آئی ٹی یونیورسٹی میں داخلے کے لیے اہلیت کا معیار یہ ہے

- میٹرک میں کم از کم 60% نمبر
- انٹرمیڈیٹ (پری انجینیئرنگ یا آئی سی ایس) میں کم از کم 65% نمبر
- داخلہ ٹیسٹ میں کم از کم 65% نمبر

ایک پروگرام لکھیں جو میٹرک، انٹرمیڈیٹ اور داخلہ ٹیسٹ کے حاصل کردہ نمبر اور کل نمبر ان پٹ لے اور بتائے کہ طالب علم اہل ہے یا نہیں۔

**مشق 7:**

ایک پروگرام لکھیں جو دیے گئے ٹیبل کی بنا پر بتائے کہ ملازم کو کتنا بونس ملے گا۔

| Salary | Experience with Company | Bonus Tasks | Bonus |
|---|---|---|---|
| 10000 | 2 Years | 5 | 1500 |
| 10000 | 3 Years | 10 | 2500 |
| 25000 | 3 Years | 4 | 2000 |
| 75000 | 4 Years | 7 | 3500 |
| 100000 | 5 Years | 10 | 5000 |

پروگرام ملازم کی تنخواہ، تجربہ اور بونس ٹاسکس کی تعداد ان پٹ لے اور سکرین پر بونس دکھائے۔

# 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- لینگویج 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 لینگویجز 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 لینگویج 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۔

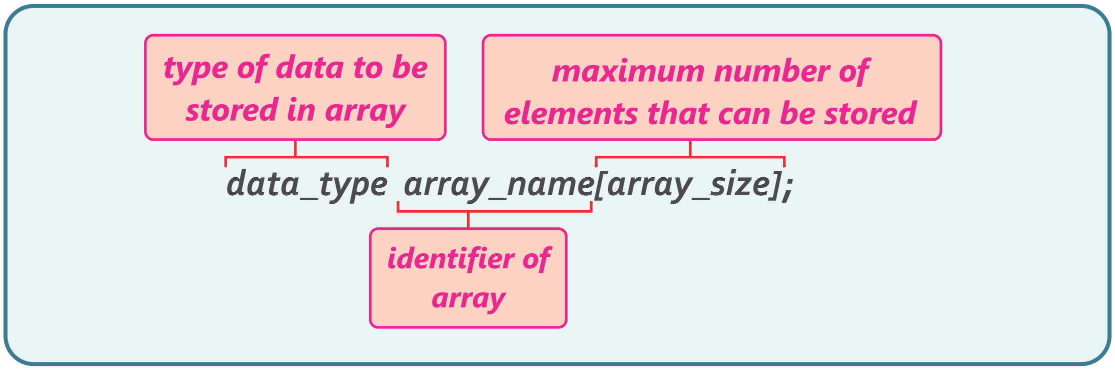
### 4.1.1 ارے (Array)

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۔


type of data to be stored in array
maximum number of elements that can be stored
data_type array_name[array_size];
identifier of array

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 انیشلائز کی گئی ہے۔

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

**اہم نوٹ:**

اگر ہم ڈکلیئریشن کے وقت ارے کو انیشلائز نہیں کرتے تو ہمیں ایک ایک کر کے ارے کے ایلیمنٹس کو انیشلائز کرنا پڑتا ہے۔ اس کا مطلب یہ کہ پھر ہم ایک سٹیٹمنٹ میں ارے کے تمام ایلیمنٹس کو انیشلائز 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};
}
```

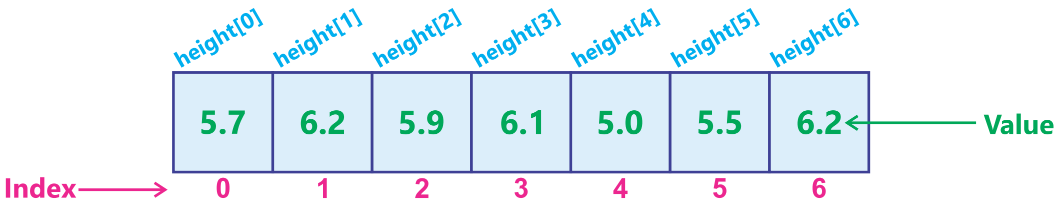
ERROR initializing whole array after declaration not allowed

اوپر دی گئی مثال میں جب ہم ڈکلیئر کرنے کے بعد ارے کو ایک الگ سٹیٹمنٹ میں انیشلائز 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 انیشلائز 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۔

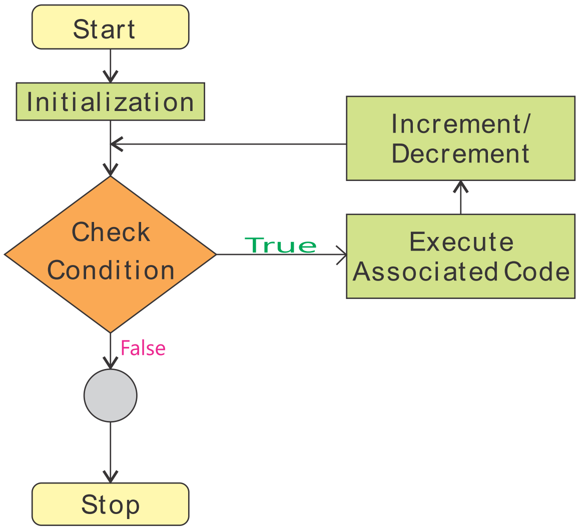
مختلف پروگرامنگ لینگویجز میں ہدایات کے ایک سیٹ کی تکرار کے لیے لوپ سٹرکچرز کا یہی فلسفہ استعمال ہوتا ہے۔

### 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۔


Start
Initialization
Increment/ Decrement
Check Condition
True
Execute Associated Code
False
Stop

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 آ جاتے 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 گے 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 کاؤنٹر ویری 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 (1<= 10) 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 ہو جائے 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 جا رہا ہے لہٰذا ہم لوپ کے ذریعے اسے حل کر سکتے 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 والاکوڈ (Code to repeat) C-لینگوئج 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 آ جاتی 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 آ کر 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 چھے 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 کاؤنٹر 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 ستعمال 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 ایلمینٹس کی ایک ارے کے تمام ایلمینٹس 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 نمبران 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 ایلیمینٹس 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- لینگوئج 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 انشلا ئزیشن کہلاتا ہے۔ ایک ارے کو ڈکلیئریشن کے وقت یا بعد میں انشلا ئز کیا جا سکتا ہے۔ ڈکلیئریشن کے ساتھ ہی ارے کو انشلا ئز 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 لینگوئج 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 کثیرالانتخابی سوالات:

1۔ ارے ایک ــــــــــــــــ سٹرکچر ہے:

a) لوپ b) کنٹرول c) ڈیٹا d) مشروط

2۔ ارے کے ایلیمنٹس میموری کے مقامات ــــــــــــــ 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 ــــــــــــــــ ہوگی:

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

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

a) لوپ b) مشروط c) کنٹرول d) ڈیٹا

5۔ ــــــــــــــــ 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 ــــــــــــ انیشلائز کیا جا سکتا ہے:

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

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

a) for b) while c) do while d) نیسٹڈ

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

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

9۔ ــــــــــــــــ 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 ــــــــــــــــ انیشلائز کریں:

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

**سوال نمب 2: درج ذیل اِصطلاحات کی تعریف کریں۔**

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

**سوال نمب 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 انیشلا ئز 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۔

# باب 5

# فنکشنز

## تدریسی مقاصد: (Student Learning OutComes)

- فنکشنز کا تصور اور ان کی اقسام کی وضاحت
- فنکشنز کے استعمال کے فوائد کی وضاحت
- فنکشنز کے سگنیچر (نام، آرگیومینٹس، 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) ویری ایبلز بطور آرگیومینٹ 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 حکمتِ 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 حکمتِ عملی کو تقسیم کرنا اور فتح کرنا (Divide and conquer) کہتے ہیں۔ C پروگرامنگ لینگویج میں ہمارے پاس فنکشنز ہوتے ہیں جو پروگرامنگ کے سوال کو حل کرنے کے لیے تقسیم کرنے اور فتح کرنے کی حکمتِ 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 انھیں کئی مرتبہ استعمال کر سکتے ہیں۔

### 5.1.1 فنکشنز کی اقسام

بنیادی طور پر فنکشنز کی دو اقسام ہیں:

1۔ بِلٹ اِن 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، اِن 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 فنکشنیلیٹی (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۔

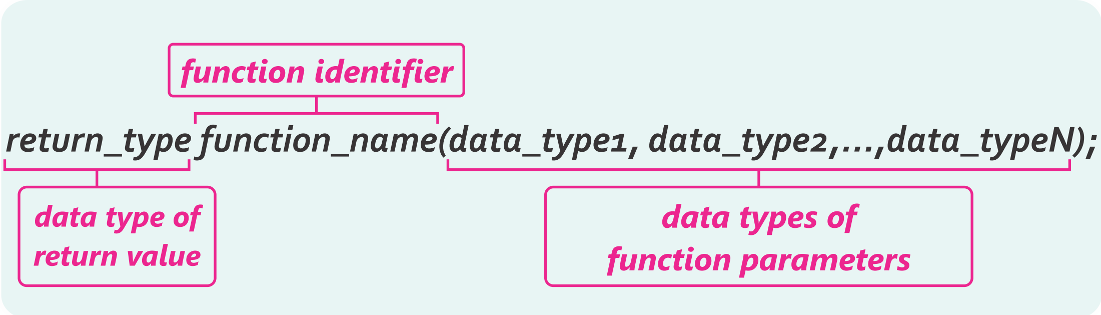
### 4۔ پڑھنے کی صلاحیت (Readability)

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 اِن پُٹ لیتا ہے۔ فنکشن کی اِن پُٹس کو پیرامیٹرز کہتے ہیں اور آؤٹ پُٹ کو ریٹرن ویلیو (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 اِن پُٹس اور آؤٹ پُٹ ڈیفائن کرتا ہے۔ فنکشن کے سگنیچر کا عام ڈھانچہ یہ ہے۔



```
return_type function_name(data_type1, data_type2,...,data_typeN);
```

### فنکشن سگنیچرز کی مثالیں:

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

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

ٹیبل 5.1 کچھ فنکشنز اور ان کے سگنیچرز

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

ایک فنکشن کے سگنیچر 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 جنھیں چلا کر فنکشن کوئی خاص کام سرانجام دیتا ہے۔ فنکشن سگنیچر کے فوراً بعد { } قوسین میں لکھی گئی سٹیٹمنٹس فنکشن کی باڈی بناتی ہیں۔ درج ذیل مثال ایک فنکشن کو ڈیفائن کرتی ہے *showPangram()* جو نہ تو کچھ اِن پُٹ 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”);
}
```

function name

چوں کہ درج بالا فنکشن کچھ ریٹرن نہیں کرتا اس لیے اس کی ریٹرن ٹائپ void ہے۔

آیئے! اب ایک اور مثال لیتے ہیں جو دو انٹیجر (integer) اِن پُٹ لے اور ان کی جمع ریٹرن کرے۔

**EXAMPLE CODE 5.2**

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

parameters of function

function name

return type

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 کرسکتا مثلاً:

درج ذیل سٹیٹمنٹ کے نتیجے میں کمپائلر ایرر دیتا ہے۔

return (4,5);

**اہم نوٹ:**

ایک فنکشن میں ایک سے زیادہ ریٹرن سٹیٹمنٹس ہوسکتی ہیں لیکن جیسے ہی ایک ریٹرن سٹیٹمنٹ چلتی ہے فنکشن کی کال ختم ہوجاتی 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:**

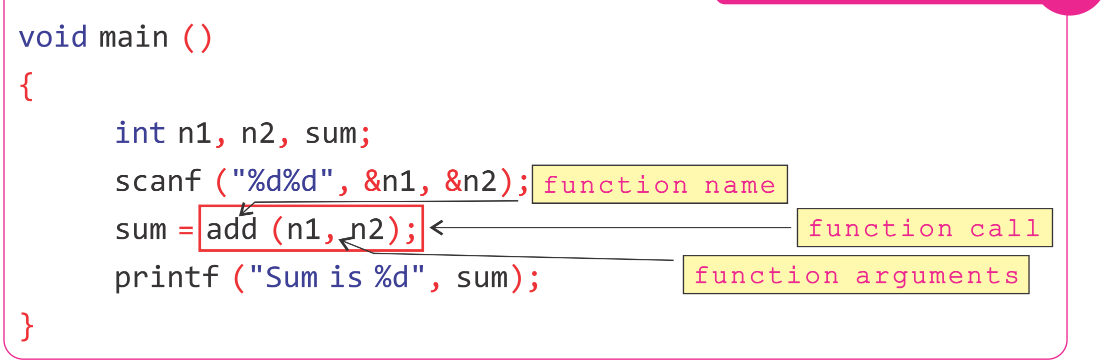
```
Hello from main()
A quick brown fox jumps over the lazy dog.
Welcome back to main()
```

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 ہوجاتا 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 اُن 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);
}
```

- فنکشن کال میں n1 اور n2 فنکشن add() کے آرگومینٹس ہیں۔
- فنکشن add() سے ریٹرن ہونے والے نتیجے کو محفوظ کرنے کے لیے متغیر sum ڈکلیئر کیا گیا ہے۔
- فنکشن کو بطور آرگومینٹ پاس کیے گئے ویری ایبلز میں کوئی تبدیلی نہیں آتی۔ فنکشن 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 فنکشن 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 فنکشن کو پاس کی جاتی ہیں وہ آرگومنٹس کہلاتی ہیں جب کہ فنکشن ڈیفینیشن میں جن ویری ایبلز میں یہ قیمتیں جاتی ہیں وہ فنکشن کے پیرامیٹرز کہلاتے ہیں۔

اُوپر دی گئی مثال میں ویری ایبلز n1 اور n2 آرگومنٹس ہیں جو فنکشن add() کو پاس
کیے گئے ہیں جبکہ فنکشن 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 فنکش کی ڈیفینیشن کال کرنے والے فنکشن کی ڈیفینیشن سے پہلے آئے تو فنکشن سگینچر 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 سگینچر اِس کال کرنے والے فنکشن سے پہلے لکھنا ضروری ہے۔

نیچے دیے گئے دونوں کوڈ سٹرکچرز 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 اِن 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)

**پرابلم:**

ایک فنکشن لکھیں جو ایک مثبت نمبر اِن 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- سٹینڈرڈ 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 سیگنیچر فنکش کے نام، ان پٹس اور آؤٹ پٹ کی وضاحت کرتا ہے۔
- فنکشن کو اس طرح ڈیفائن کیا جا سکتا 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 جنھیں 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 آرگیومینٹس 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۔سٹینڈرڈ لائبریری میں موجود فنکشنز _______________ کہلاتے ہیں:

الف) یوزر ڈیفائنڈ ب) بلٹ ان ج) تکرار پر مبنی د) تکراری

3) فنکشن کو پاس کی گئی قیمتیں ____________ کہلاتی ہیں:

الف) باڈیز ب) ریٹرن ٹائپس ج) ارے د) آرگیومنٹس

4) char cd() {return = 'a';} اس فنکشن میں "char" ____________ ہے:

الف) باڈی ب) ریٹرن ٹائپ ج) ارے د) آرگیومینٹس

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 باڈی 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) آرگیومینٹس 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 = xyx(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 نمبر1:**

ایک انٹیجر (integer) x کا مربع معلوم کرنے کے لیے `int square (int x);` فنکشن لکھیں۔

**مشق نمبر2:**

ایک فنکشن `int power(int x, int y);` لکھیں جو $x^y$ معلوم کرکے ریٹرن کرے۔

**مشق نمبر3:**

ایک نمبر کا فیکٹوریل نکالنے کا فنکشن لکھیں۔

**مشق نمبر4:**

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 نمبر5:**

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

**مشق نمبر6:**

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

**مشق نمبر7:**

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۔

**آرگیومینٹس:** 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 انیشلائزیشن:** پہلی مرتبہ ارے میں قیمتیں لکھنا۔ ایک ارے کو ڈکلیریشن کے وقت یا اس کے بعد انیشلائز کیا جاسکتا ہے۔

**ارے کا سائز:** زیادہ سے زیادہ ایلیمنٹس 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 اوپریٹر:** 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 فنکشنیلٹی منسوب ہو۔

**انڈیکس:** یہ ارے کے ایلیمنٹس تک رسائی حاصل کرنے کے لیے استعمال ہوتا ہے۔ متغیرات کو بھی ارے انڈیکسز کے طور پر استعمال کیا جاسکتا ہے۔

**انٹیجر ڈیٹا ٹائپ:** ایک ڈیٹا ٹائپ جس میں انٹیجر کانسٹینٹس محفوظ کیے جاتے ہیں۔ اس کا سائز 4 بائٹس ہے۔

**انٹیگریٹڈ ڈیویلپمنٹ انوائرمنٹ:** ایک ایسا سافٹ ویئر جو پروگرامر کو کمپیوٹر پروگرام لکھنے اور چلانے کے لیے پروگرامنگ انوائرمنٹ فراہم کرے۔

**if سٹیٹمنٹ:** یہ کنڈیشن سے منسوب کوڈ تب 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 اوپریٹر، ریلیشنل اوپریٹر اور منطقی اوپریٹر شامل ہیں۔

**بائنری اوپریٹرز:** انھیں دو اوپرینڈز 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 انوائرمنٹ:** پروگرامنگ کے تمام اہم آلات کا مجموعہ۔

**پروگرامنگ لینگویجز:** 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 اوپرینڈ درکار ہوتے 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 کرکے ان کا تعلق بتاتے ہیں۔

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

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

**سٹیٹمنٹ ٹرمینیٹر:** ایک شناخت کنندہ جو کمپائلر کو بتاتا ہے کہ سٹیٹمنٹ ختم ہوگئی ہے۔ C- لینگوئج میں سیمی کولن (;) بطور سٹیٹمنٹ ٹرمینیٹر استعمال ہوتا ہے۔

**سٹرنگ:** کریکٹر کا مجموعہ۔

**سنٹیکس:** ہر پروگرامنگ لینگوئج کے کچھ ابتدائی تعمیراتی عناصر اور پروگرام لکھنے کے کچھ اصول ہوتے ہیں۔ اصولوں کے اس سیٹ کو لینگوئج 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 باڈی:** یہ ان سٹیٹمنٹس کا سیٹ ہوتا ہے۔ جنھیں چلا کر فنکشن مخصوص کام سرانجام دیتا ہے۔

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

**فلوٹنگ پوائنٹ:** ایک ڈیٹا ٹائپ جو چھے ہندسوں تک پر سائز رئیل کانسٹنٹ (Precise Real Constant) محفوظ کرنے کے لیے استعمال ہوتی ہے۔ اس کا سائز 4 بائٹس ہے۔

**فارمیٹ سپیسفائرز:** یہ ان پٹ آؤٹ پٹ اوپریشنز کے دوران ڈیٹا ٹائپ کا فارمیٹ بنانے کے لیے استعمال ہوتے ہیں۔

**فنکشن سگنیچر:** یہ فنکشن کی ان پٹ اور آؤٹ پٹ کی وضاحت کرتا ہے۔

**فنکشن:** سٹیٹمینٹس 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۔

**کمنٹس:** وہ سٹیٹمنٹس جو چلتی نہیں ہیں۔

**کمپائلر:** ایک ایسا سافٹ ویئر جو کسی ہائی لیول کی پروگرامنگ لینگوئج 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 ارتھمیٹک ایکسپریشنز، ریلیشنل ایکسپریشنز، منطقی ایکسپریشنز یا ان سب کا مجموعہ۔

**کانسٹینٹس:** وہ ڈیٹا جسے مزید پروسیسنگ کے لئے کمپیوٹر کی میموری میں محفوظ کیا جاتا ہے۔

**کنٹرول سٹیٹمنٹس:** وہ پروگرام جو تسلسل کو کنٹرول کرتی ہے۔

**کی۔ورڈز:** پروگرامنگ لینگوئج میں پہلے سے ڈیفائن کیے ہوئے الفاظ کی فہرست۔

**ہیڈر فائلز:** وہ فائلز جن میں ڈیزائنرز نے موجودہ فنکشنز ڈیفائن کیے ہوں۔

**ہیڈر سیکشن:** وہ سیکشن جس میں ہیڈر فائلز شامل کی جاتی ہیں۔

**لوپ سٹرکچر:** یہ سٹیٹمنٹس کے ایک سیٹ کو بار بار دہرانے کے لیے استعمال ہوتا ہے۔

**مشروط سٹیٹمنٹس:** وہ سٹیٹمینٹس جو شرائط کی بنا پر یہ فیصلہ کرنے میں مدد دیتی ہیں کے آگے کونسی سٹیٹمینٹس چلانی ہیں۔

**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 اوپریشن انجام دیتے ہیں اور جواب میں بولین قیمت واپس کرتے ہیں۔

**مین فنکشن:** یہاں سے پروگرام چلنا شروع ہوتا ہے۔

**ماڈولس اوپریٹر:** ایک بائنری اوپریٹر جو بائیں اوپرینڈ کو دائیں اوپرینڈ پر تقسیم کرتا ہے اور تقسیم کے بعد بچنے والی رقم واپس کرتا ہے۔

**متغیر کی ڈکلیریشن:** متغیر کا نام اور ڈیٹا ٹائپ متعین کرنا۔

**متغیر کی انیشلائزیشن:** پہلی مرتبہ ایک متغیر سے قیمت منسوب کرنا۔

**متغیرات:** وہ کنٹینرز جن میں کانسٹینٹس یا قیمتیں محفوظ کی جاتی ہیں۔

**نیسٹڈ سلیکشن سٹرکچرز:** مشروط سٹیٹمنٹس میں مشروط سٹیٹمنٹس

**printf()** فارمیٹڈ آؤٹ پٹ سکرین پر دکھانے کا بلٹ ان فنکشن۔

**scanf():** صارف سے فارمیٹڈ ان پٹ ریڈ کرنے والا C لینگوئج کا بلٹ ان فنکشن۔

**یونری اوپریٹرز:** انھیں صرف ایک اوپرینڈ درکار ہوتا ہے۔

**یوزر ڈیفائنڈ فنکشنز:** وہ فنکشنز جنھیں پروگرامر خود ڈیفائن کرے۔

# انڈیکس

(الف)
AND اوپریٹر، 39
آرگومینٹس، 108
ارتھمیٹک اوپریٹرز، 32
ارے کی ڈکلریشن، 79
ارے کی انیشلائزیشن، 79
انڈیکس، 80
انٹیجر کانسٹینٹ، 10
انٹیجر، 12
انٹی گریٹڈ ڈویلپمنٹ انوائرنمنٹ، 3
اوپریٹر، 31
ان سائنڈ int، 12
OR اوپریٹر، 40
ارے کا سائز، 79
ارے، 78
اسائنمنٹ اوپریٹر، 316
ایڈیٹر، 4
اسکیپ کریکٹر، 29
ان پٹ، آؤٹ پٹ اوپریشنز، 23
if سٹیٹمنٹس، 53
if-else سٹیٹمنٹس، 59
(ب)
بائنری اوپریٹر، 41
بولین، 12
بلٹ ان فنکشن، 103
بار بار استعمال کی صلاحیت، 104
(پ)
پیرامیٹرز، 104
Printf' فنکشن، 23
پروگرام، 2
پروگرامنگ انوائرنمنٹ، 2
پروگرامنگ لینگوئجز، 2
(ت)
ترجیح، 41
(ج)
جمع کا اوپریٹر، 35
(ڈ)
ڈیٹا سٹرکچر، 78
ڈیٹا ٹائپ، 116
(ذ)
ذخیرہ الفاظ، 6
(ر)
رئیل کانسٹنٹ، 10
ریشنل کریکٹرز، 37
ریٹرن ویلیو، 104
(س)
Scanf، 26
سائنڈ int، 12
سنگل لائن کمنٹ، 8
سٹیٹمنٹ ٹرمینیٹر، 29
سٹرنگ، 12
سنٹیکس ایرر، 5
سنٹیکس، 5
(ش)
شناخت کنندہ، 11
شرط 53
(ض)
ضرب کا آپریٹرز، 34
(ف)
فلوٹنگ پوائنٹ، 12
فارمیٹ سپیسفائر، 24
فنکشن کال، 107
فنکشن ڈیفینیشن، 105
فنکشن سگنیچر، 105
فنکشن، 103
(ک)
کریکٹر کانسٹینٹ، 34
کنٹرول سٹیٹمنٹس، 52
کمپیوٹر پروگرام، 2
کمپیوٹر پروگرام، 2
conio.h، 28
کنسول، 5
کانسٹینٹس، 10
(گ)
getch()، 28
(ل)
لنک سیکشن، 6
لوپ سٹرکچر، 83
(م)
منسوب کردہ کوڈ، 53
مین فنکشن کی باڈی، 7
مطلوبہ الفاظ، 6
منطقی اوپریٹرز، 39
مین فنکشن، 7
مین فکشن 7
ماڈولس آپریٹر، 36

# جوابات

## باب 1:

### سوال نمبر 1: کثیر الانتخابی سوالات۔

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

### سوال نمبر 2: درست/غلط۔

1۔ درست
2۔ درست
3۔ غلط
4۔ غلط
5۔ درست

### سوال نمبر 3: کالم ملائیں۔

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

# جوابات

## باب 2:

### سوال نمبر 1: کثیر الانتخابی سوالات۔

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

### سوال نمبر 2: درست/غلط۔

1۔ غلط
2۔ درست
3۔ غلط
4۔ غلط
5۔ غلط

### سوال نمبر 3: آؤٹ پٹ۔

1۔ 0 7 9
2۔ nn

```
nnn
 n
t     nn /n/n nn/n
```

3۔ 9
4۔ 5
5۔ 1

## 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

## باب4:

سوال نمبر1: کثیر الانتخابی سوالات۔

1۔ ج
2۔ الف
3۔ الف
4۔ الف
5۔ ب

6۔ د
7۔ د
8۔ ج
9۔ الف
10۔ الف

سوال نمبر5: آؤٹ پٹ۔

**1.** Sum is 25

**2.** *

**3.**
```
j = 50
j = 49
j = 48
i = 50
```

**4.**
```
4
16
36
64
```

**5.**
```
0.000000
0.000000
0.000000
2.640000
5.520000
11.959999
```

## باب5:

سوال نمبر1: کثیر الانتخابی سوالات۔

1۔ ج
2۔ ب
3۔ د
4۔ ب

5۔ د
6۔ الف
7۔ د
8۔ الف

سوال نمبر5: آؤٹ پٹ۔

1۔ 20
2۔ 4 7 23
3۔ We got 260
4۔
```
0.000000
0.000000
```