REVIEW OF C PROGRAMMING

Instructor: Dr. Sukhjit Singh Sehra

Overview of C Programming

- Example: Basic C program structure

```c
#include <stdio.h>

int main() {
printf("Hello, World!\n");
return 0;
}
```

The C Program Lifecycle

- Example: Simple program to demonstrate compiling and running

```c
#include <stdio.h>

int main() {
printf("Welcome to C Programming!\n");
return 0;
}
```

First C Program

- Example: Detailed “Hello, World!” program with comments

```c
// Include the standard input/output library
#include <stdio.h>

// Main function - Execution starts here
int main() {
// Print "Hello, World!" to the console
printf("Hello, World!\n");
// Return 0 to indicate successful completion
return 0;
}
```

Variables and Assignments

Understanding Variables and Data Types

- Example: Declaring different data types
-

```c
int age = 30; // Integer
float height = 5.8; // Floating-point number
char grade = 'A';// Character
```

Variable Operations

- Example: Demonstrating variable initialization and scope

```c
#include <stdio.h>

int globalVar = 10; // Global variable

int main() {
int localVar = 5; // Local variable
printf("Global variable: %d\n", globalVar);
printf("Local variable: %d\n", localVar);
return 0;
}
```

Variables in Action

- Example: Using `sizeof()` and variable scope

```c
#include <stdio.h>

int main() {
int a = 5;
printf("Size of int: %zu bytes\n", sizeof(a));
return 0;
}
```

Branches (Conditional Statements)

Introduction to Conditional Branching

- Example: Basic `if` statement

```c
#include <stdio.h>

int main() {
int number = 10;
if (number > 0) {
 printf("The number is positive.\n");
}
return 0;
}
```

Advanced Conditional Branching

- Example: Using `else if` and `else`

```c
#include <stdio.h>

int main() {
int number = -5;
if (number > 0) {
 printf("Positive\n");
} else if (number < 0) {
 printf("Negative\n");
} else {
 printf("Zero\n");
}
return 0;
}
```

The `switch` Statement

- Example: Implementing `switch`

```c
#include <stdio.h>

int main() {
char grade = 'B';
switch (grade) {
 case 'A':
printf("Excellent!\n");
break;
 case 'B':
printf("Well done\n");
break;
 case 'C':
printf("You passed\n");
break;
 case 'D':
printf("Better try again\n");
break;
 default:
printf("Invalid grade\n");
}
return 0;
}
```

Loops

Introduction to Loops

- Example: Basic `for` loop

```c
#include <stdio.h>

int main() {
for (int i = 1; i <= 5; i++) {
 printf("%d ", i);
}
return 0;
}
```

While and Do-While Loops

- Example: Using a `while` loop

```c
#include <stdio.h>

int main() {
int i = 1;
while (i <= 5) {
 printf("%d ", i);
 i++;
}
return 0;
}
```

While and Do-While Loops

- Example: `do-while` loop

```c
#include <stdio.h>

int main() {
int i = 1;
do {
printf("%d ", i);
i++;
} while (i <= 5);
return 0;
}
```

Loop Control Statements

- Example: Using `break` and `continue`

```c
#include <stdio.h>

int main() {
for (int i = 1; i <= 10; i++) {
if (i == 6) {
break; // exit loop when i is 6
```

```
}
if (i % 2 == 0) {
continue; // skip the current iteration if i is even
}
printf("%d ", i);
}
return 0;
}
```

Arrays

Introduction to Arrays

- Example: Declaring and accessing arrays

```
#include <stdio.h>

int main() {
int numbers[5] = {1, 2, 3, 4, 5};
for (int i = 0; i < 5; i++) {
printf("%d ", numbers[i]);
}
return 0;
}
```

Multi-dimensional Arrays

- Example: Using 2D arrays

```
#include <stdio.h>

int main() {
int matrix[2][3] = {{1, 2, 3}, {4, 5, 6}};
for (int i = 0; i < 2; i++) {
for (int j = 0; j < 3; j++) {
printf("%d ", matrix[i][j]);
}
printf("\n");
}
return 0;
}
```

User-defined Functions

Basics of Functions

- Example: Defining and calling a function

```c
#include <stdio.h>

int add(int a, int b) {
return a + b;
}

int main() {
int result = add(5, 3);
printf("Result: %d\n", result);
return 0;
}
```

Function Parameters and Return Values

- Example: Passing parameters by value

```c
#include <stdio.h>

void update(int a) {
a = 10;
}

int main() {
int a = 5;
update(a);
printf("a after update: %d\n", a); // a remains 5
return 0;
}
```

Advanced Function Concepts

- Example: Recursive function

```c
#include <stdio.h>

int factorial(int n) {
if (n <= 1) {
return 1;
} else {
return n * factorial(n - 1);
}
}

int main() {
int result = factorial(5);
printf("Factorial of 5: %d\n", result);
return 0;
}
```

Structs

Introduction to Structs

- Example: Defining and using structs

```
#include <stdio.h>

struct Person {
char name[50];
int age;
};

int main() {
struct Person person1 = {"Alice", 30};
printf("%s is %d years old.\n", person1.name, person1.age);
return 0;
}
```

Structs Cond...

- Example: Arrays of structs

```
#include <stdio.h>

struct Point {
int x, y;
};

int main() {
struct Point points[2] = {{1, 2}, {3, 4}};
for (int i = 0; i < 2; i++) {
printf("Point %d: (%d, %d)\n", i, points[i].x, points[i].y);
}
return 0;
}
```

Structs Cond...

- Example: Passing structs to functions

```
#include <stdio.h>

struct Person {
char name[50];
int age;
};

void printPerson(struct Person p) {
```

```
printf("Name: %s, Age: %d\n", p.name, p.age);
}

int main() {
struct Person person1 = {"Bob", 25};
printPerson(person1);
return 0;
}
```

Practical Use Cases for Structs

- Example: Nested structs

```
#include <stdio.h>

struct Date {
int day, month, year;
};

struct Person {
char name[50];
struct Date birthday;
};

int main() {
struct Person person1 = {"Alice", {12, 10, 1990}};
printf("%s was born on %d/%d/%d\n", person1.name, person1.birthday.day, person1.birthday.mon
return 0;
}
```

Pointers

Understanding Pointers

- Example: Basic pointer usage

```
#include <stdio.h>

int main() {
int var = 20;
int *ptr = &var;

printf("Address of var: %p\n", &var);
printf("Address stored in ptr: %p\n", ptr);
printf("Value of *ptr: %d\n", *ptr);
return 0;
}
```

Pointers in Action

- Example: Pointer arithmetic

```
#include <stdio.h>

int main() {
int arr[] = {10, 20, 30, 40, 50};
int *ptr = arr;

for (int i = 0; i < 5; i++) {
 printf("arr[%d] = %d\n", i, *(ptr + i));
}
return 0;
}
```

Pointers in Action

- Example: Pointers and arrays

```
#include <stdio.h>

int main() {
int arr[] = {10, 20, 30, 40, 50};
int *ptr = arr;

for (int i = 0; i < 5; i++) {
 printf("arr[%d] = %d\n", i, ptr[i]); // ptr[i] is equivalent to *(ptr + i)
}
return 0;
}
```

Pointers with Functions and Structs

- Example: Passing pointers to functions

```
#include <stdio.h>

void doubleValue(int *num) {
*num *= 2;
}

int main() {
int value = 5;
doubleValue(&value);
printf("Doubled value: %d\n", value);
return 0;
}
```

- Example: Pointers in structs

```c
#include <stdio.h>

struct Point {
int x, y;
};

void printPoint(struct Point *p) {
printf("(%d, %d)\n", p->x, p->y);
}

int main() {
struct Point point1 = {10, 20};
printPoint(&point1);
return 0;
}
```

Input/Output

Basic I/O in C

- Example: Using `printf` and `scanf`

```c
#include <stdio.h>

int main() {
int number;
printf("Enter a number: ");
scanf("%d", &number);
printf("You entered: %d\n", number);
return 0;
}
```

File I/O Operations

- Example: Writing to a file

```c
#include <stdio.h>

int main() {
FILE *filePtr;
filePtr = fopen("example.txt", "w");
if (filePtr == NULL) {
 printf("Error opening file!\n");
 return 1;
}
fprintf(filePtr, "Hello, file!\n");
fclose(filePtr);
return 0;
}
```

Reading from a File

- Example: Reading from a file

```
#include <stdio.h>

int main() {
FILE *filePtr;
char buffer[100];
filePtr = fopen("example.txt", "r");
if (filePtr == NULL) {
printf("Error opening file!\n");
return 1;
}
while (fgets(buffer, 100, filePtr) != NULL) {
printf("%s", buffer);
}
fclose(filePtr);
return 0;
}
```

Recursion

Introduction to Recursion

- Example: Concept of Recursion

```
#include <stdio.h>

void countDown(int num) {
if (num == 0) {
printf("Blastoff!\n");
} else {
printf("%d\n", num);
countDown(num - 1); // Recursive call
}
}

int main() {
countDown(5);
return 0;
}
```

Recursive Functions in Practice

- Example: Factorial using Recursion

```
#include <stdio.h>
```

```
int factorial(int n) {
if (n <= 1) {
return 1;
} else {
return n * factorial(n - 1); // Recursive call
}
}

int main() {
int num = 5;
printf("Factorial of %d is %d\n", num, factorial(num));
return 0;
}
```

Understanding Recursion with Fibonacci Series

- Example: Fibonacci Series using Recursion

```
#include <stdio.h>

int fibonacci(int n) {
if (n <= 1) {
return n;
} else {
return fibonacci(n - 1) + fibonacci(n - 2); // Recursive calls
}
}

int main() {
int terms = 10;
printf("First %d terms of Fibonacci series:\n", terms);
for (int i = 0; i < terms; i++) {
printf("%d ", fibonacci(i));
}
printf("\n");
return 0;
}
```