Structures in C

A structure is a user-defined data type that groups related variables of different types.

Syntax:

```
struct StructureName {
datatype member1;
datatype member2;
};
```

Example:

```
struct Student {
int id;
char name[50];
};

struct Student s1 = {101, "Alice"};
printf("ID: %d", s1.id);
```

Pointers in C

A pointer is a variable that stores the memory address of another variable.

Syntax:

datatype *pointer_name;

Example:

int a = 10;

int *ptr = &a;

printf("Address of a: %p", ptr);

printf("Value of a: %d", *ptr);

Pointer Arithmetic:

- ptr++, ptr--
- *ptr: dereferencing

Functions in C

Functions are blocks of code that perform a specific task.

Syntax:

```
return_type function_name(parameters) {
// code
}
```

Example:

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

int main() {
int sum = add(5, 3);
printf("Sum: %d", sum);
return 0;
}
```

Types of Functions:

- Library Functions (e.g., printf, scanf)
- User-defined Functions

Arrays in C

An array is a collection of similar data elements stored under a single variable name.

Syntax:

datatype array_name[size];

Example:

int numbers[5] = {1, 2, 3, 4, 5};

Accessing elements:

printf("%d", numbers[2]); // Output: 3

Types of Arrays:

1. One-dimensional array

2. Two-dimensional array (matrix)

3. Multi-dimensional array

Control Structures in C

Control structures determine the flow of control in a program.

1. Conditional Statements:

- if, if-else, else-if ladder

- switch

Example:

```
if (age >= 18) {
printf("Adult");
} else {
printf("Minor");
}
```

2. Loops:

- for loop

- while loop

- do-while loop

Example:

```
for (int i = 0; i < 5; i++) {
printf("%d\n", i);
}
```

3. Jump Statements:

- break, continue, goto, return

Operators and Expressions in C

Operators are symbols that perform operations on variables and values.

Types of Operators:

1. Arithmetic Operators: +, -, *, /, %

2. Relational Operators: ==, !=, >, <, >=, <=

3. Logical Operators: &&, ||, !

4. Assignment Operators: =, +=, -=, etc.

5. Increment/Decrement: ++, --

Example:

```
int a = 5, b = 2;

int sum = a + b; // sum = 7

int product = a * b; // product = 10
```

Input and Output in C

The standard library provides functions for input and output.

Output using printf():

```
#include <stdio.h>
int main() {
printf("Welcome to C Programming!");
return 0;
}
```

Input using scanf():

```
#include <stdio.h>
int main() {
int age;
printf("Enter your age: ");
scanf("%d", &age);
printf("You are %d years old", age);
return 0;
}
```

Format Specifiers:

- %d: integer
- %f: float
- %c: character
- %s: string

Variables and Data Types in C

A variable is a name given to a memory location. It is used to store data.

Syntax:

datatype variable_name;

Example:

int age;

float salary;

Common Data Types:

- int: stores integers (e.g., 1, -5, 100)

- float: stores decimal numbers (e.g., 3.14, -0.99)

- char: stores single characters (e.g., 'a', 'Z')

- double: stores large decimal numbers

Variable Declaration and Initialization:

int age = 25;

float pi = 3.14;

char grade = 'A';

Introduction to C Language

C is a powerful general-purpose programming language developed by Dennis Ritchie in the early 1970s. It is widely used for system programming, developing operating systems, and embedded systems.

Features of C:

- Fast and efficient
- Structured programming language
- Rich set of built-in operators and functions
- Portable and flexible

Example:

```
#include <stdio.h>

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