Programming in C (4331105) - Summer 2023 Solution

Milav Dabgar

July 26, 2023

Question 1 [a marks]

3 List any six keywords of C language.

Solution

Table: Six Keywords in C Language

Keyword	Purpose
`int`	Integer data type
`float`	Floating-point data type
`if`	Conditional statement
`while`	Loop structure
`return`	Returns value from function
`void`	Specifies empty return type

Mnemonic

"I Feel When Running Very Ill" (int, float, while, return, void, if)

Question 1 [b marks]

4 Define variable. List the rule for naming of variable in c programming.

Solution

Variable: A named memory location used to store data that can be modified during program execution.

Table: Rules for Variable Naming in C

Rule	Example
Must begin with letter/underscore	`name`, `_value`
Can contain letters, digits, underscore	`user_1`, `count99`
No spaces or special characters	✓: `total_sum`, ✗: `total-sum`
Case sensitive	`Name` ≠ `name`
Cannot use reserved keywords	✗: `int`, `while`
Maximum 31 characters (standard)	`studentRegistrationNumber`

Mnemonic

"Letters Lead, No Special Keys" (begins with letter, no special chars, no keywords)

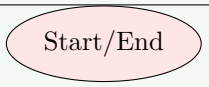
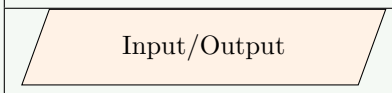
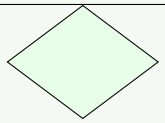
Question 1 [c marks]

7 Define flowchart. Draw and Explain flowchart symbols. Write a program to calculate simple interest using below equation. I=PRN/100 Where P=Principal amount, R=Rate of interest and N=Period.

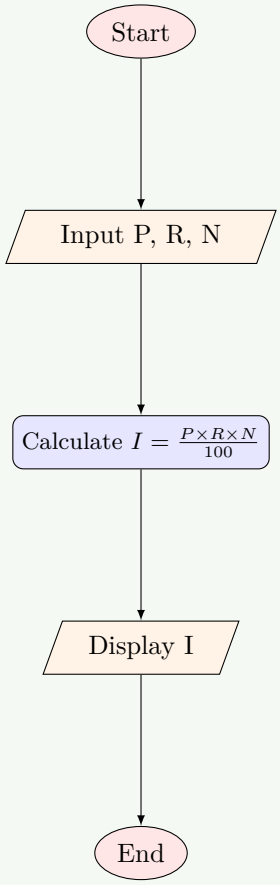
Solution

Flowchart: A graphical representation of an algorithm that uses standard symbols to show the sequence of operations needed to solve a problem.

Table: Flowchart Symbols

Symbol	Name	Purpose
	Terminal	Start/End
	Process	Calculations
	Input/Output	Read/Display data
	Decision	Conditions
→	Flow Line	Shows sequence

Simple Interest Flowchart:



Program:

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

```c
void main()
{
    float p, r, n, i;

    printf("Enter principal amount: ");
    scanf("%f", &p);

    printf("Enter rate of interest: ");
    scanf("%f", &r);

    printf("Enter time period in years: ");
    scanf("%f", &n);

    i = (p * r * n) / 100;

    printf("Simple Interest = %.2f", i);
}
```

Mnemonic

"Please Return Nice Interest" (Principal, Rate, Number of years, Interest)

Question 1 [c marks]

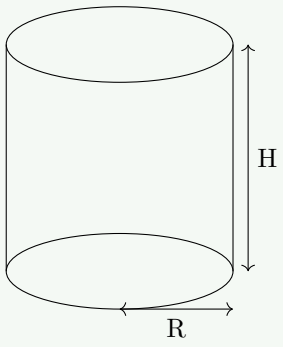
7 OR Define algorithm. Write algorithm for finding volume of cylinder. Write a program to read radius(R) and height(H) from user and print calculated the volume(V) of cylinder using. V=$\pi R^2 H$.

Solution

Algorithm: A step-by-step procedure to solve a problem in a finite amount of time.
Algorithm for Cylinder Volume:

1. Start
2. Input radius (R) and height (H)
3. Calculate volume using formula $V = \pi \times R^2 \times H$
4. Display the volume
5. End

Diagram: Cylinder



Program:

```c
#include <stdio.h>
void main()
{
    float radius, height, volume;
    float pi = 3.14159;
```

```c

    printf("Enter radius of cylinder: ");
    scanf("%f", &radius);

    printf("Enter height of cylinder: ");
    scanf("%f", &height);

    volume = pi * radius * radius * height;

    printf("Volume of cylinder = %.2f", volume);
}
```

Mnemonic

"Round Hat Volume" (Radius, Height, Volume)

Question 2 [a marks]

3 List out different operators supported in C programming language.

Solution

Table: Operators in C Programming

Operator Type	Examples	Use
Arithmetic	+, -, *, /, %	Mathematical operations
Relational	<, >, ==, !=, <=, >=	Compare values
Logical	&&, ||, !	Combine conditions
Assignment	=, +=, -=, *=, /=	Assign values
Increment/Decrement	++, –	Increase/decrease by 1
Bitwise	&, |, ^, ~, «, »	Bit manipulation
Conditional	?:	Short if-else

Mnemonic

"All Relationships Lead Ancestors Incrementally Beyond Conditions" (first letter of each type)

Question 2 [b marks]

4 Write a program to print sum and average of 1 to 50.

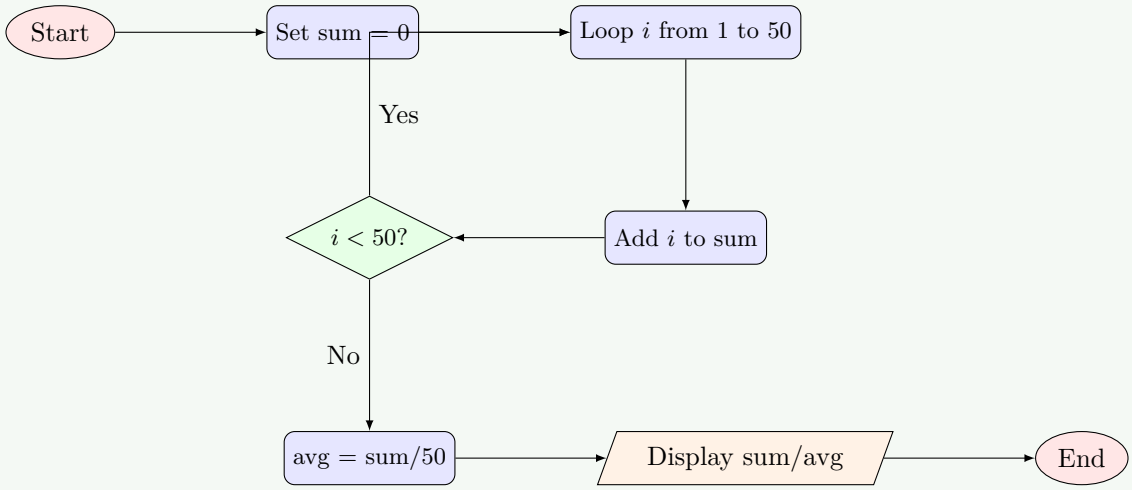
Solution

Program:

```c
#include <stdio.h>
void main()
{
    int i, sum = 0;
    float avg;

    for(i = 1; i <= 50; i++)
    {
```

```c
        sum = sum + i;
    }

    avg = (float)sum / 50;

    printf("Sum of numbers from 1 to 50 = %d\n", sum);
    printf("Average of numbers from 1 to 50 = %.2f", avg);
}
```

Process Diagram:



Mnemonic

"Summing And Dividing" (Sum, Average, Division)

Question 2 [c marks]

7 Explain arithmetic & relational operators with example.

Solution

Arithmetic Operators:

Table: Arithmetic Operators in C

Operator	Operation	Example	Result
+	Addition	5 + 3	8
-	Subtraction	7 - 2	5
*	Multiplication	4 * 3	12
/	Division	8 / 4	2
%	Modulus (Remainder)	7 % 3	1

Relational Operators:

Table: Relational Operators in C

Operator	Meaning	Example	Result
<	Less than	5 < 8	1 (true)
>	Greater than	9 > 3	1 (true)
==	Equal to	4 == 4	1 (true)
!=	Not equal to	7 != 3	1 (true)
<=	Less than/equal	4 <= 4	1 (true)
>=	Greater than/equal	6 >= 9	0 (false)

Code Example:

```c
#include <stdio.h>
void main()
{
    int a = 10, b = 5;

    // Arithmetic operators
    printf("a + b = %d\n", a + b);   // 15
    printf("a - b = %d\n", a - b);   // 5
    printf("a * b = %d\n", a * b);   // 50
    printf("a / b = %d\n", a / b);   // 2
    printf("a %% b = %d\n", a % b);  // 0

    // Relational operators
    printf("a < b: %d\n", a < b);    // 0 (false)
    printf("a > b: %d\n", a > b);    // 1 (true)
    printf("a == b: %d\n", a == b);  // 0 (false)
    printf("a != b: %d\n", a != b);  // 1 (true)
}
```

Mnemonic

"Add Subtract Multiply Divide Remainder" (arithmetic), "Less Greater Equal Not" (relational)

Question 2 [a marks]

3 OR State the difference between gets(S) and scanf("%s",S) where S is string.

Solution

Table: Difference between gets(S) and scanf("%s",S)

Feature	gets(S)	scanf("%s",S)
Space handling	Reads spaces between words	Stops reading at space
Input termination	Ends at newline character	Ends at whitespace
Buffer overflow	Unsafe, no length check	Safer with width limit
Example behavior	"Hello World" → "Hello World"	"Hello World" → "Hello"
Security	Deprecated due to overflow risks	Better with width specifier

Mnemonic

"Gets Spaces, Scanf Stops" (gets reads spaces, scanf stops at spaces)

Question 2 [b marks]

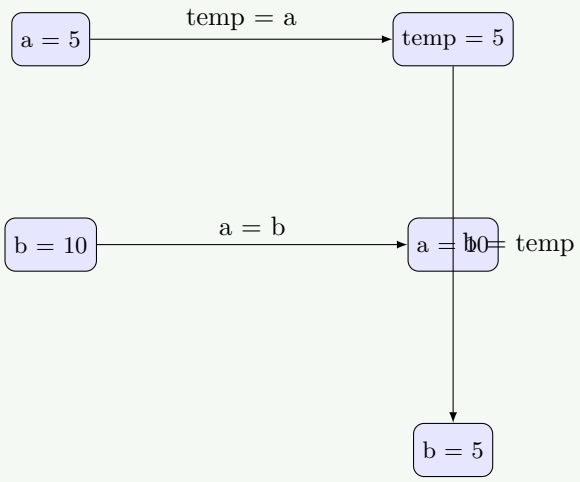
4 OR Write a program to swap two numbers.

Solution

Program:

```c
#include <stdio.h>
void main()
{
    int a, b, temp;

    printf("Enter value of a: ");
    scanf("%d", &a);

    printf("Enter value of b: ");
    scanf("%d", &b);

    printf("Before swapping: a = %d, b = %d\n", a, b);

    // Swapping using temp variable
    temp = a;
    a = b;
    b = temp;

    printf("After swapping: a = %d, b = %d", a, b);
}
```

Swapping Diagram:



Mnemonic

"Temporary Assists Swapping" (Temp variable enables swapping)

Question 2 [c marks]

7 OR Explain Logical operator and bit-wise operator with example.

Solution

Logical Operators:

Table: Logical Operators in C

Operator	Description	Example	Result
&&	Logical AND	(5>3) && (8>6)	1 (both true)
||	Logical OR	(5<3) || (8>6)	1 (one true)
!	Logical NOT	!(5>3)	0 (inverts true)

Bitwise Operators:

Table: Bitwise Operators in C

Operator	Description	Example	Binary Result
&	Bitwise AND	5 & 3	101 & 011 = 001 (1)
|	Bitwise OR	5 | 3	101 | 011 = 111 (7)
^	Bitwise XOR	5 ^ 3	101 ^ 011 = 110 (6)
~	Bitwise NOT	~5	~0101 = 1010 (-6)
«	Left Shift	5 « 1	101 « 1 = 1010 (10)
»	Right Shift	5 » 1	101 » 1 = 10 (2)

Code Example:

```c
#include <stdio.h>
void main()
{
    int a = 5, b = 3;

    // Logical operators
    printf("a>3 && b<5: %d\n", (a>3) && (b<5));  // 1 (true)
    printf("a<3 || b>1: %d\n", (a<3) || (b>1));  // 1 (true)
    printf("!(a>b): %d\n", !(a>b));              // 0 (false)

    // Bitwise operators
    printf("a & b: %d\n", a & b);   // 1
    printf("a | b: %d\n", a | b);   // 7
    printf("a ^ b: %d\n", a ^ b);   // 6
    printf("~a: %d\n", ~a);         // -6
    printf("a << 1: %d\n", a << 1); // 10
    printf("a >> 1: %d\n", a >> 1); // 2
}
```

Mnemonic

"AND OR NOT" (logical), "AND OR XOR NOT SHIFT" (bitwise)

Question 3 [a marks]

3 Explain multiple if-else statement with example.

Solution

Multiple if-else: Series of if-else statements where each condition is checked sequentially until a true condition is found.

Structure:

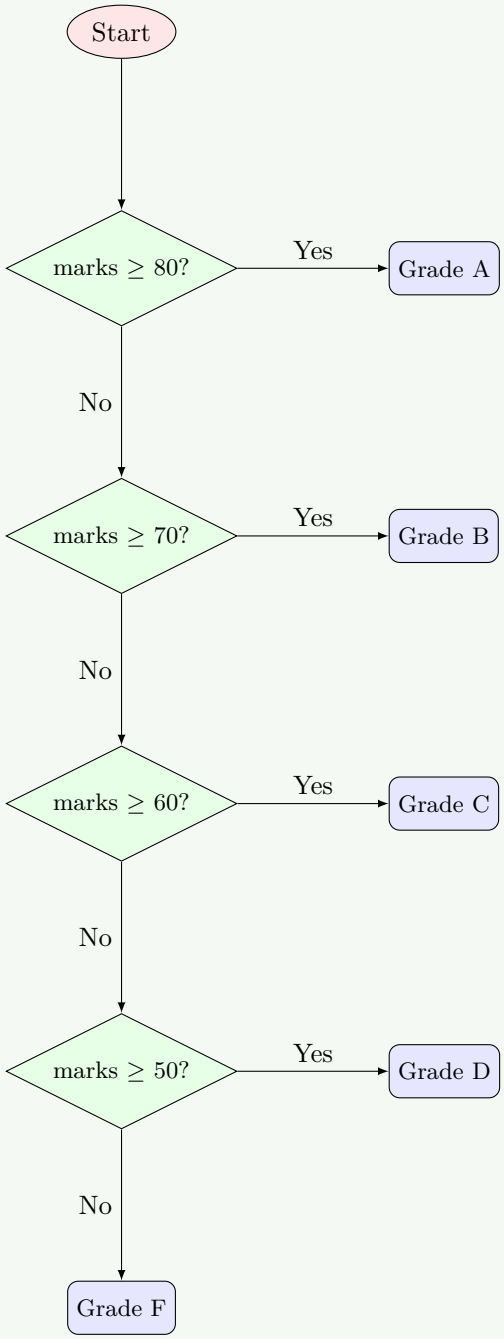
```c
if (condition1)
    statement1;
else if (condition2)
```

```
    statement2;
else if (condition3)
    statement3;
else
    default_statement;
```

Code Example:

```c
#include <stdio.h>
void main()
{
    int marks;

    printf("Enter marks: ");
    scanf("%d", &marks);

    if (marks >= 80)
        printf("Grade: A");
    else if (marks >= 70)
        printf("Grade: B");
    else if (marks >= 60)
        printf("Grade: C");
    else if (marks >= 50)
        printf("Grade: D");
    else
        printf("Grade: F");
}
```

Diagram:



Mnemonic

"Check Each Condition in Sequence" (CECS)

Question 3 [b marks]

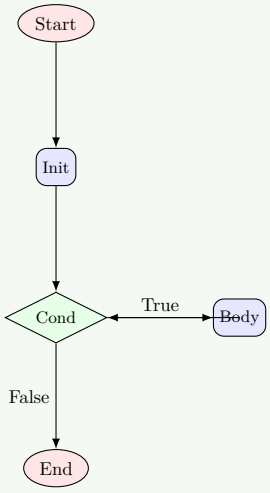
4 State the working of while loop and for loop.

Solution

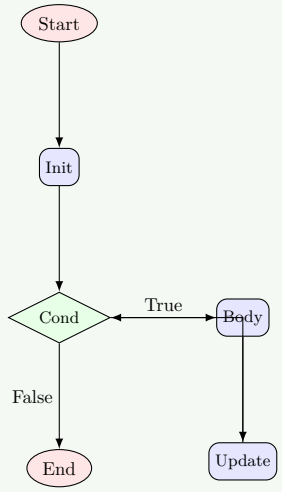
Table: While Loop vs For Loop

Feature	While Loop	For Loop
Syntax	`while(cond) { stmt; }`	`for(init; cond; upd) { stmt; }`
When to use	Iterations unknown	Iterations known
Initialization	Before loop	Inside declaration
Update	Inside loop body	In declaration
Exit control	Beginning	Beginning
Example	User input check	Fixed count

While Loop Flow:



For Loop Flow:



Mnemonic

"While Checks Then Acts" (WCTA), "For Initializes Tests Updates" (FITU)

Question 3 [c marks]

7 Write a program to find factorial of a given number.

Solution

Program:

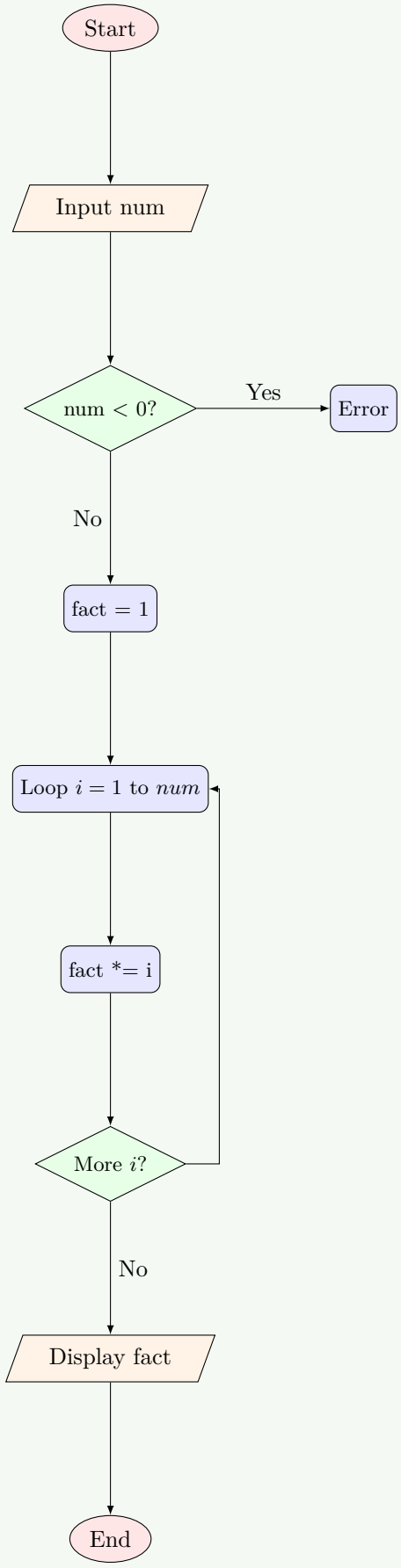
```c
#include <stdio.h>
void main()
{
    int num, i;
    unsigned long fact = 1;

    printf("Enter a number: ");
    scanf("%d", &num);

    if (num < 0)
        printf("Factorial not defined for negative numbers");
    else
    {
        for(i = 1; i <= num; i++)
        {
            fact = fact * i;
        }
        printf("Factorial of %d = %lu", num, fact);
    }
```

```
}
```

Factorial Calculation Table: For example, if num = 5:

Iter	i	fact * i	New fact
Init	-	-	1
1	1	1 * 1	1
2	2	1 * 2	2
3	3	2 * 3	6
4	4	6 * 4	24
5	5	24 * 5	120

Factorial Calculation Diagram:


Start
Input num
num < 0?
Yes
Error
No
fact = 1
Loop $i = 1$ to num
fact *= i
More i?
No
Display fact
End

Mnemonic

"Find And Count The Numbers!" (FACTN! - Factorial)

Question 3 [a marks]

3 OR Explain the working of switch-case statement with example.

Solution

Switch-Case: A selection statement that allows a variable to be tested for equality against a list of values (cases).
Structure:

```
switch(expression) {
    case value1:
        statements1;
        break;
    case value2:
        statements2;
        break;
    default:
        default_statements;
}
```

Code Example:

```
#include <stdio.h>
void main()
{
    int day;
    printf("Enter day number (1-7): ");
    scanf("%d", &day);

    switch(day) {
        case 1: printf("Monday"); break;
        case 2: printf("Tuesday"); break;
        case 3: printf("Wednesday"); break;
        case 4: printf("Thursday"); break;
        case 5: printf("Friday"); break;
        case 6: printf("Saturday"); break;
        case 7: printf("Sunday"); break;
        default: printf("Invalid day");
    }
}
```

Mnemonic

"Select Value, Exit with Break" (SVEB)

Question 3 [b marks]

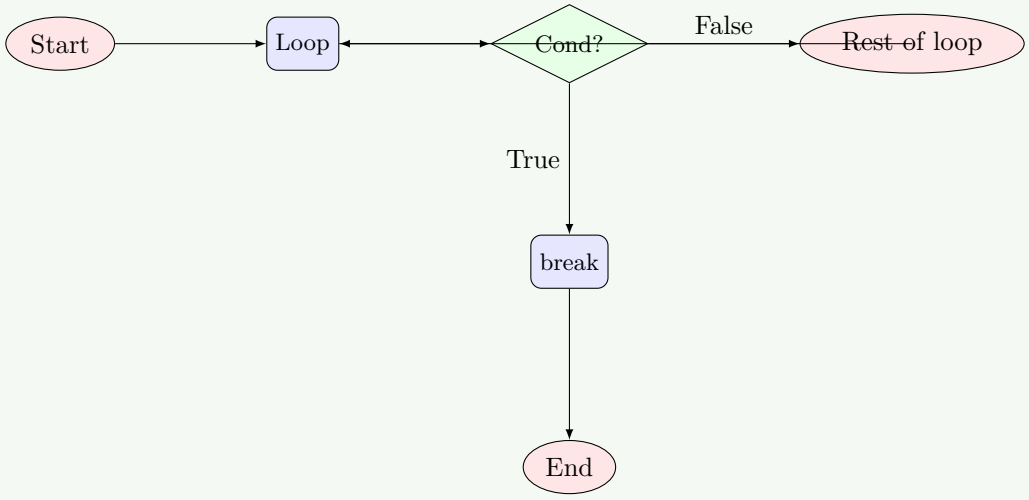
4 OR State the use of break and continue keyword.

Solution

Table: Break vs Continue Keywords

Feature	break	continue
Purpose	Exits loop/switch	Skips iteration
Effect	Terminates loop	Next iteration
Where used	Loops & switch	Loops only
Flow	After loop	Condition check

Flow Diagram - break:



Mnemonic

"Break Exits, Continue Skips" (BECS)

Question 3 [c marks]

7 OR Write a program to read number of lines (n) from keyboard and print the triangle shown below. For Example, n=5

Solution

Target Pattern:

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Program:

```c
#include <stdio.h>
void main()
{
    int n, i, j;

    printf("Enter number of lines: ");
    scanf("%d", &n);

    for(i = n; i >= 1; i--)
```

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

Pattern Logic Table: For n = 5:

i	j range	Output
5	1 to 5	1 2 3 4 5
4	1 to 4	1 2 3 4
3	1 to 3	1 2 3
2	1 to 2	1 2
1	1 to 1	1

Mnemonic

"Decreasing Rows With Increasing Values" (DRWIV)

Question 4 [a marks]

3 Explain nested if-else statement with example.

Solution

Nested if-else: An if-else statement inside another if or else block.
Structure:

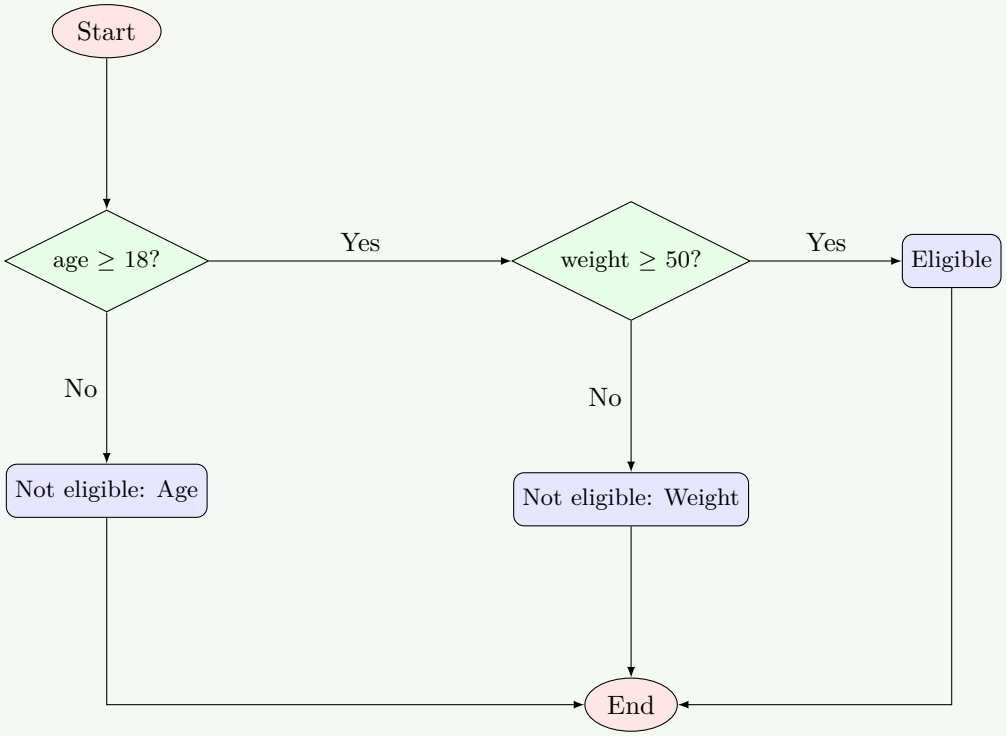
```
if (condition1) {
    if (condition2) {
        statements1;
    } else {
        statements2;
    }
} else {
    statements3;
}
```

Code Example:

```
#include <stdio.h>
void main()
{
    int age, weight;

    printf("Enter age: ");
    scanf("%d", &age);

    if (age >= 18) {
        printf("Enter weight: ");
        scanf("%d", &weight);

        if (weight >= 50) {
            printf("Eligible to donate blood");
        } else {
```

```
            printf("Underweight, not eligible");
        }
    } else {
        printf("Age below 18, not eligible");
    }
}
```

Nested if-else Diagram:



Mnemonic

"Check Outside Then Inside" (COTI)

Question 4 [b marks]

4 Write a program to exchange two integer numbers using pointer arguments.

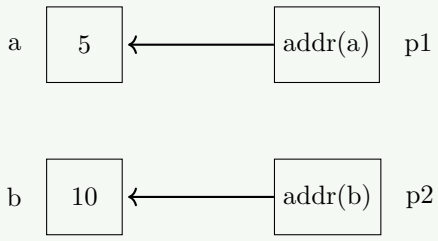
Solution

Program:

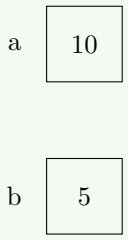
```
#include <stdio.h>
void main()
{
    int a, b, temp;
    int *p1, *p2;

    printf("Enter value of a: ");
    scanf("%d", &a);

    printf("Enter value of b: ");
    scanf("%d", &b);

```

```c
    p1 = &a;  // p1 points to a
    p2 = &b;  // p2 points to b

    printf("Before swapping: a = %d, b = %d\n", a, b);

    // Swapping using pointers
    temp = *p1;
    *p1 = *p2;
    *p2 = temp;

    printf("After swapping: a = %d, b = %d", a, b);
}
```

Pointer Swapping Diagram:



After swapping values via pointers:



Mnemonic

"Pointers Exchange Memory Values" (PEMV)

Question 4 [c marks]

7 Define Array. Explain initialization & declaration of one-dimensional array.

Solution

Array: A collection of elements of the same data type stored in contiguous memory locations and accessed using indices.

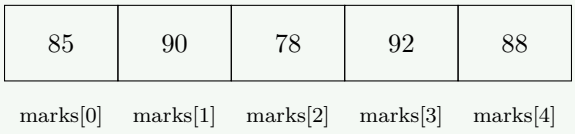
Table: Array Declaration & Initialization

Operation	Syntax	Example
Declaration	`type name[size];`	`int marks[5];`
Init at decl	`type name[size] = {vals};`	`int nums[4] = {10, 20};`
Partial init	`type name[size] = {vals};`	`int nums[5] = {10};`
No size	`type name[] = {vals};`	`int nums[] = {1, 2};`
Individual	`name[index] = value;`	`marks[0] = 95;`

Code Example:

```c
#include <stdio.h>
void main()
{
    // Declaration
    int marks[5];

    // Initialization after declaration
    marks[0] = 85; marks[1] = 90;
    marks[2] = 78; marks[3] = 92; marks[4] = 88;

    // Declaration with initialization
    int scores[] = {95, 89, 76, 82, 91};

    printf("marks[2] = %d\n", marks[2]);
}
```

Memory Representation:



Mnemonic

"Declare, Initialize, Access With Index" (DIAWI)

Question 4 [a marks]

3 OR Explain do while loop with example.

Solution

do-while loop: A loop that executes the body at least once before checking the condition.
Structure:

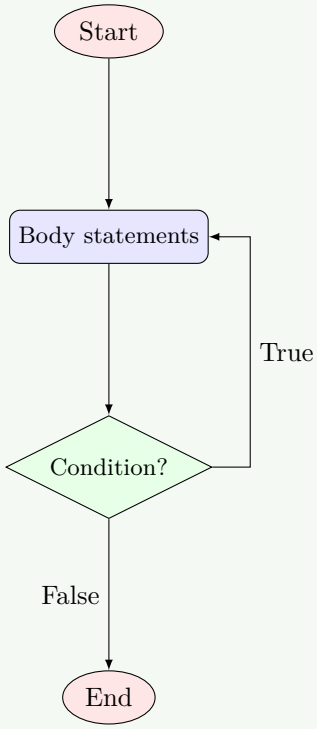
```c
do {
    statements;
} while(condition);
```

Code Example:

```c
#include <stdio.h>
void main()
{
    int num, sum = 0;
    do {
        printf("Enter a number (0 to stop): ");
        scanf("%d", &num);
```

```
        sum += num;
    } while(num != 0);
    printf("Sum = %d", sum);
}
```

do-while Loop Flow:



Key Differences from while loop:

- Body executes at least once
- Condition checked after execution
- Semicolon required after condition

Mnemonic

"Do First, Check Later" (DFCL)

Question 4 [b marks]

4 OR Explain following functions with example: (1) gets() (2) puts() (3) strlen() (4) strcpy()

Solution

Table: String Functions in C

Function	Purpose	Example
gets()	Reads string with spaces	`gets(name);`
puts()	Displays string + newline	`puts(name);`
strlen()	Returns string length	`n = strlen(str);`
strcpy()	Copies src to dest	`strcpy(d, s);`

Code Example:

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

```c
#include <string.h>
void main()
{
    char name[50], copy[50];
    int length;

    printf("Enter name: ");
    gets(name);
    puts(name);

    length = strlen(name);
    printf("Length: %d\n", length);

    strcpy(copy, name);
    printf("Copied: %s", copy);
}
```

Mnemonic

"Gets Puts String's Length and Copies" (GPSLC)

Question 4 [c marks]

7 OR Define recursion and explain with suitable example. Write a program to find factorial of a given number using recursion.

Solution

Recursion: A process where a function calls itself directly or indirectly until a specific condition is met.
Components:
1. Base case: Condition to stop recursion.
2. Recursive case: Function calling itself.

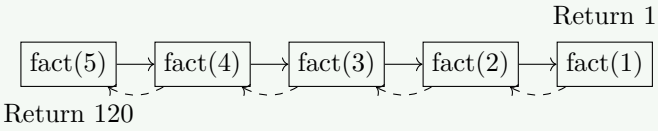
Code Example:

```c
#include <stdio.h>

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

void main()
{
    int num;
    printf("Enter a number: ");
    scanf("%d", &num);
    printf("Factorial of %d = %lu", num, factorial(num));
}
```

Recursion Trace for factorial(5):

Call	Step	Result
fact(5)	5× fact(4)	$5 \times 24 = 120$
fact(4)	4× fact(3)	$4 \times 6 = 24$
fact(3)	3× fact(2)	$3 \times 2 = 6$
fact(2)	2× fact(1)	$2 \times 1 = 2$
fact(1)	Base case	1

Recursion Diagram:



Mnemonic

"Function Calling Itself, Bottoming Out" (FCIBO)

Question 5 [a marks]

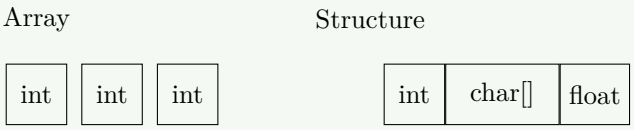
3 Write the difference between array and structure.

Solution

Table: Array vs Structure

Feature	Array	Structure
Data type	Same for all elements	Different types
Access	Index (`arr[0]`)	Member (`s.name`)
Memory	Contiguous	Contiguous (mixed)
Size	Fixed	Sum of members
Purpose	Collection of similar	Grouping related
Decl	`int a[5];`	`struct s { int a; };`

Diagram:



Mnemonic

"Arrays for Same, Structures for Different" (ASSD)

Question 5 [b marks]

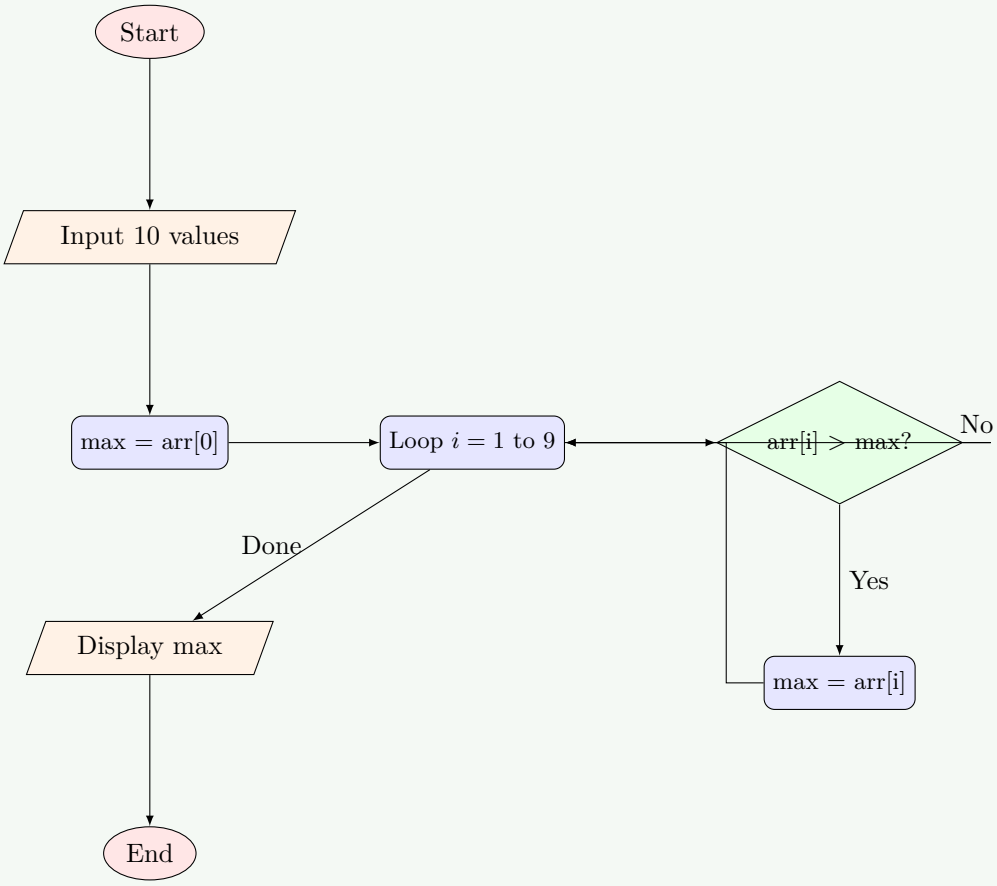
4 Write a C program using array that find the maximum value from given 10 values.

Solution

Program:

```c
#include <stdio.h>
void main()
{
    int arr[10], i, max;

    printf("Enter 10 values:\n");
    for(i = 0; i < 10; i++) {
        scanf("%d", &arr[i]);
    }

    max = arr[0];
    for(i = 1; i < 10; i++) {
        if(arr[i] > max)
            max = arr[i];
    }

    printf("Maximum value is: %d", max);
}
```

Algorithm Flow:



Mnemonic

"Compare And Replace Maximum" (CARM)

Question 5 [c marks]

7 Define structure? Develop a structure named book to save following information about books. Book

title, Name of author, Price and Number of pages.

Solution

Structure: A user-defined data type that groups related variables of different data types under a single name.
Book Structure Code:

```c
#include <stdio.h>

struct book {
    char title[50];
    char author[30];
    float price;
    int pages;
};

void main()
{
    struct book b1;

    printf("Enter book title: "); gets(b1.title);
    printf("Enter author name: "); gets(b1.author);
    printf("Enter price: "); scanf("%f", &b1.price);
    printf("Enter pages: "); scanf("%d", &b1.pages);

    printf("\nBook Details:\n");
    printf("Title: %s\n", b1.title);
    printf("Author: %s\n", b1.author);
    printf("Price: %.2f\n", b1.price);
    printf("Pages: %d", b1.pages);
}
```

Structure Diagram:

struct book
title[50] (char)
author[30] (char)
price (float)
pages (int)

Mnemonic

"Title Author Price Pages" (TAPP)

Question 5 [a marks]

3 OR What is a string? What are the operations that can be performed on string?

Solution

String: A sequence of characters terminated by a null character '\0'.
Table: String Operations in C

Operation	Function	Example
Input	gets, scanf	`gets(s)`
Output	puts, printf	`puts(s)`
Length	strlen	`l=strlen(s)`
Copy	strcpy	`strcpy(d,s)`
Concat	strcat	`strcat(s1,s2)`
Compare	strcmp	`strcmp(s1,s2)`
Search	strchr	`strchr(s,'a')`

String Representation:

H	e	l	l	o	\0

Mnemonic

"Input Output Length Copy Concat Compare Search Convert" (IOLCCSC)

Question 5 [b marks]

4 OR Write a program prints its ASCII value from A to Z.

Solution

Program:

```c
#include <stdio.h>
void main()
{
    char ch;

    printf("ASCII values from A to Z:\n");
    printf("Char\tValue\n");

    for(ch = 'A'; ch <= 'Z'; ch++)
    {
        printf("%c\t%d\n", ch, ch);
    }
}
```

ASCII Representation:

A (65)	B (66)	...	Z (90)

Mnemonic

"Alphabets Sequentially Creating Integer Indices" (ASCII)

Question 5 [c marks]

7 OR What is user defined and library function? Explain with two examples of each.

Solution

Library Functions: Pre-defined functions provided by C language that are ready to use (e.g., printf, sqrt).
User-Defined Functions: Functions created by the programmer to perform specific tasks.

Table: Library vs User-Defined Functions

Feature	Library Functions	User-Defined Functions
Definition	Pre-defined	By programmer
Decl	Not needed	Required
Header	Required (stdio.h)	Not required
Purpose	Common tasks	Custom tasks

Examples:

1. **Library**: `strlen("Hi")` (string.h), `sqrt(25)` (math.h)
2. **User-Defined**:

```c
float calculateArea(float l, float w) {
    return l * w;
}
int findMax(int a, int b) {
    return (a>b)?a:b;
}
```

Mnemonic

"Libraries Provide, Users Create" (LPUC)