

EXPERIMENT12: PREPROCESSOR AND DIRECTIVES IN C

Experiment 1: Defining Constants Using Preprocessor Directives

Aim

To define and use constant variables in C using preprocessor directives.

Algorithm

1. Use #define to declare constants
2. Read required input values
3. Perform calculations using constants
4. Display results

Pseudocode

```
DEFINE PI = 3.14
READ radius
area = PI * radius * radius
DISPLAY area
```

Flowchart (ASCII)

```
Start
↓
Define Constant
↓
Read Input
↓
Perform Calculation
↓
Display Result
↓
End
```

C Program

```
#include <stdio.h>

#define PI 3.14

int main() {
    float r, area;

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

    area = PI * r * r;

    printf("Area of circle = %.2f\n", area);
    return 0;
}
```

Output

```
PS C:\Users\ASUS\Desktop\Dahadi\class_c> gcc t.c
PS C:\Users\ASUS\Desktop\Dahadi\class_c> ./a.exe
Enter radius: 4
Area of circle = 50.24
PS C:\Users\ASUS\Desktop\Dahadi\class_c>
```

Conclusion

Preprocessor directive #define was successfully used to declare constants in the program.

Experiment 2 : Defining Function Using Preprocessor Directives

Aim

To define and use a macro function using preprocessor directives in C.

Algorithm

1. Define macro function using #define
2. Read input numbers
3. Apply macro function
4. Display result

Pseudocode

```
DEFINE MAX(a, b)
READ a, b
result = MAX(a, b)
DISPLAY result
```

Flowchart (ASCII)

```
Start
↓
Define Macro Function
↓
Read Inputs
↓
Apply Macro
↓
Display Result
↓
End
```

C Program

```
#include <stdio.h>

#define MAX(a, b) ((a > b) ? a : b)

int main() {
    int x, y;

    printf("Enter two numbers: ");
```

```
    scanf("%d %d", &x, &y);

    printf("Maximum value = %d\n", MAX(x, y));

    return 0;
}
```

Output

```
PS C:\Users\ASUS\Desktop\Dahadi\class c> gcc t.c
PS C:\Users\ASUS\Desktop\Dahadi\class c> ./a.exe
Enter two numbers: 34
1000
Maximum value = 1000
PS C:\Users\ASUS\Desktop\Dahadi\class c> █
```

Conclusion

Macro functions defined using preprocessor directives execute faster as they avoid function call overhead.