1. Write a C program using expressions with assignment operators.

#include <stdio.h>

int main()

{

int a = 5, c;

c = a; // c is 5

printf("c = %d\n", c);

c += a; // c is 10

printf("c = %d\n", c);

c -= a; // c is 5

printf("c = %d\n", c);

c \*= a; // c is 25

printf("c = %d\n", c);

c /= a; // c is 5

printf("c = %d\n", c);

c %= a; // c = 0

printf("c = %d\n", c);

return 0;

}

Output:

c = 5

c = 10

c = 5

c = 25

c = 5

c = 0

1. Explain in detail about the essential steps for problem solving?

**Program Analysis**

• Understand the problem

• Have multiple solutions

• Select a solution

**Program Design**

• Write Algorithm

• Write Flowchart

•Write Pseudo code

**Program Development**

• Choose a programming language

• Write the program by converting the pseudo code, and using the

programming language.

• Compile the program and remove syntax errors, if any

• Execute the program.

• Test the program. Check the output results with different inputs. If the

output is incorrect, modify the program to get correct results.

• Install the tested program on the user’s computer.

**Program Documentation and**

**maintenance**

• Document the program, for later use.

• Maintain the program for updating, removing errors, changing

requirements etc.

1. Explain about C Token.

We can define the token as the smallest individual element in C. For `example, we cannot create a sentence without using words; similarly, we cannot create a program in C without using tokens in C. Therefore, we can say that tokens in C is the building block or the basic component for creating a program in C language.

Classification of tokens in C

Tokens in C language can be divided into the following categories:

Keywords in C

Identifiers in C

Strings in C

Operators in C

Constant in C

Special Characters in C

Refer <https://www.javatpoint.com/tokens-in-c>

1. Write a C program using expressions with relational operators.

#include <stdio.h>

int main()

{

int a = 5, b = 5, c = 10;

printf("%d == %d is %d \n", a, b, a == b);

printf("%d == %d is %d \n", a, c, a == c);

printf("%d > %d is %d \n", a, b, a > b);

printf("%d > %d is %d \n", a, c, a > c);

printf("%d < %d is %d \n", a, b, a < b);

printf("%d < %d is %d \n", a, c, a < c);

printf("%d != %d is %d \n", a, b, a != b);

printf("%d != %d is %d \n", a, c, a != c);

printf("%d >= %d is %d \n", a, b, a >= b);

printf("%d >= %d is %d \n", a, c, a >= c);

printf("%d <= %d is %d \n", a, b, a <= b);

printf("%d <= %d is %d \n", a, c, a <= c);

return 0;

}

Output:

5 == 5 is 1

5 == 10 is 0

5 > 5 is 0

5 > 10 is 0

5 < 5 is 0

5 < 10 is 1

5 != 5 is 0

5 != 10 is 1

5 >= 5 is 1

5 >= 10 is 0

5 <= 5 is 1

5 <= 10 is 1

1. Write a c program using expressions with logical operators.

#include <stdio.h>

int main()

{

int a = 5, b = 5, c = 10, result;

result = (a == b) && (c > b);

printf("(a == b) && (c > b) is %d \n", result);

result = (a == b) && (c < b);

printf("(a == b) && (c < b) is %d \n", result);

result = (a == b) || (c < b);

printf("(a == b) || (c < b) is %d \n", result);

result = (a != b) || (c < b);

printf("(a != b) || (c < b) is %d \n", result);

result = !(a != b);

printf("!(a != b) is %d \n", result);

result = !(a == b);

printf("!(a == b) is %d \n", result);

return 0;

}

Output:

(a == b) && (c > b) is 1

(a == b) && (c < b) is 0

(a == b) || (c < b) is 1

(a != b) || (c < b) is 0

!(a != b) is 1

!(a == b) is 0

1. Write a program with expressions using bitwise operators.

#include <stdio.h>

int main()

{

// a = 5(00000101), b = 9(00001001)

unsigned char a = 5, b = 9;

// The result is 00000001

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

printf("a&b = %d\n", a & b);

// The result is 00001101

printf("a|b = %d\n", a | b);

// The result is 00001100

printf("a^b = %d\n", a ^ b);

// The result is 11111010

printf("~a = %d\n", a = ~a);

// The result is 00010010

printf("b<<1 = %d\n", b << 1);

// The result is 00000100

printf("b>>1 = %d\n", b >> 1);

return 0;

}

Output:

a = 5, b = 9

a & b = 1

a | b = 13

a ^ b = 12

~a = -6

b << 1 = 18

b >> 1 = 4

1. Write a C program with expressions using arithmetic operators.

#include <stdio.h>

int main()

{

int a = 9,b = 4, c;

c = a+b;

printf("a+b = %d \n",c);

c = a-b;

printf("a-b = %d \n",c);

c = a\*b;

printf("a\*b = %d \n",c);

c = a/b;

printf("a/b = %d \n",c);

c = a%b;

printf("Remainder when a divided by b = %d \n",c);

return 0;

}

1. Write a C program with different format specifiers.

#include <stdio.h>

main() {

   char ch = 'B';

   printf("%c", ch); //printing character data

   //print decimal or integer data with d and i

   int x = 45, y = 90;

   printf("%d", x);

   printf("%i", y);

   float f = 12.67;

   printf("%f", f); //print float value

   printf("%e", f); //print in scientific notation

   int a = 67;

   printf("%o", a); //print in octal format

   printf("%x", a); //print in hex format

   char str[] = "Hello World";

   printf("%s", str);

   printf("%20s", str); //shift to the right 20 characters including the string

   printf("%-20s", str); //left align

   printf("%20.5s", str); //shift to the right 20 characters including the string, and print string up to 5 character

   printf("%-20.5s", str); //left align and print string up to 5 character

}

Output

B

45

90

12.670000

1.267000e+001

103

43

Hello World

Hello World

Hello World

Hello

Hello

1. Write a C program with escape sequences.

|  |  |
| --- | --- |
| **Escape Sequence** | **Meaning** |
| \a | Alarm or Beep |
| \b | Backspace |
| \f | Form Feed |
| \n | New Line |
| \r | Carriage Return |
| \t | Tab (Horizontal) |
| \v | Vertical Tab |
| \\ | Backslash |
| \' | Single Quote |
| \" | Double Quote |
| \? | Question Mark |
| \nnn | octal number |
| \xhh | hexadecimal number |
| \0 | Null |

#include <stdio.h>

**int** main()

{

printf("Hello\nWorld!"); //use of \n

printf("\nHello\tWorld!"); // use of \t

printf("\n\"Hello World!\""); //use of \"

printf("\nHello\bWorld!"); //use of \b

**return** 0;

}

Output

Hello

World!

Hello World!

"Hello World!"

HellWorld!

1. What is an ASCII value. Write a program to print an ASCII value of a character.

The full form of ASCII is the **American Standard Code for information interchange**. It is a character encoding scheme used for electronics communication. Each character or a special character is represented by some ASCII code, and each ascii code occupies 7 bits in memory.

In [C programming language](https://www.javatpoint.com/c-programming-language-tutorial), a character variable does not contain a character value itself rather the ascii value of the character variable. The ascii value represents the character variable in numbers, and each character variable is assigned with some number range from 0 to 127. For example, the ascii value of 'A' is 65.

#include <stdio.h>

**int** main()

{

**char** ch;    // variable declaration

    printf("Enter a character");

    scanf("%c",&ch);  // user input

    printf("\n The ascii value of the ch variable is : %d", ch);

**return** 0;

}

Output

Enter a character

A

The asci value of ch variable is 65.

9. What are the formatted input/ output functions? Explain. What are the unformatted input/ output functions? Explain.

**Formatted Input and Output:**

**Scanf**

The scanf() function is used for inputs formatted from standard inputs and provides numerous conversion options for the printf() function.

Syntax:

scanf(format\_specifiers, &data1, &data2,……); // & is address operator

The scanf() function reads and converts the characters from the standard input according to the format specification string and stores the input in the memory slots represented by the other arguments.

**Printf**

The printf() function is used for output formatted as the standard output according to a format specification. The format specification string and the output data are the parameters of the printf() function.

Syntax:

printf(format\_specifiers, data1, data2,…..... );

**sprintf:**

sprintf stands for “string print”. This function is similar to printf() function but this function prints the string into a character array instead of printing it on the console screen.

Syntax:

sprintf(array\_name, “format specifier”, variable\_name);

**sscanf:**

sscanf stands for “string scanf”. This function is similar to scanf() function but this function reads data from the string or character array instead of the console screen.

Syntax:

sscanf(array\_name, “format specifier”, &variable\_name);

**Unformatted Input and Output:**

These functions are called unformatted I/O functions because we cannot use format specifiers in these functions and hence, cannot format these functions according to our needs.

The following unformatted I/O functions will be discussed in this section-

getch()

getche()

getchar()

putchar()

gets()

puts()

putch()

getch():

getch() function reads a single character from the keyboard by the user but doesn’t display that character on the console screen and immediately returned without pressing enter key. getch() is also used for hold the screen.

Syntax:

getch();

or

variable-name = getch();

getchar():

The getchar() function is used to read only a first single character from the keyboard whether multiple characters is typed by the user and this function reads one character at one time until and unless the enter key is pressed. This function is declared in stdio.h(header file)

Syntax:

Variable-name = getchar();

putchar():

The putchar() function is used to display a single character at a time by passing that character directly to it or by passing a variable that has already stored a character. This function is declared in stdio.h(header file)

Syntax:

putchar(variable\_name);

gets():

gets() function reads a group of characters or strings from the keyboard by the user and these characters get stored in a character array. This function allows us to write space-separated texts or strings. This function is declared in stdio.h(header file).

Syntax:

char str[length of string in number]; //Declare a char type variable of any length

gets(str);

puts():

In C programming puts() function is used to display a group of characters or strings which is already stored in a character array. This function is declared in stdio.h(header file).

Syntax:

puts(identifier\_name );

putch():

putch() function is used to display a single character which is given by the user and that character prints at the current cursor location. This function is declared in conio.h(header file)

Syntax:

putch(variable\_name);