# Keywords, Identifier, Literals, Operators and Expression Assignment

**Mandatory:**

1. Choose all valid identifiers

Ans) In C, an identifier is a name used to identify variables, functions, arrays, or any other user-defined item. It consists of letters (a-z, A-Z), digits (0-9), and underscores (\_), but it cannot start with a digit. Identifiers are case-sensitive and must not be a reserved keyword in C.

* 1. int int 🡪 invaild
  2. int \_numvalue 🡪valid
  3. float price\_money 🡪valid
  4. char name1234567890123456789012345678901234567890 🡪valid
  5. char name valu e🡪 invaild
  6. char $name 🡪 invaild

1. What is the meaning of the following keywords, show the usage
   1. Auto

* In C, auto is used to declare local variables with automatic storage duration. However, it's largely redundant in modern C because local variables are automatically considered auto by default.
* Usage:

#include <stdio.h>

int main() {

auto int x = 5;

printf("%d\n", x);

return 0;

}

* 1. Extern
* extern is used to declare a variable or function that is defined outside the current file or scope, typically in another file. It tells the compiler that the variable or function is defined elsewhere, so it should not allocate memory for it again.
* Usage:

#include <stdio.h>

extern int x;

int main() {

printf("%d\n", x);

return 0;

}

* 1. Volatile
* volatile tells the compiler not to optimize or cache the value of a variable, as it might be changed by something outside the program (e.g., hardware or an interrupt). It is typically used for hardware-related or multithreaded applications where a variable might change unexpectedly.
* Usage:

#include <stdio.h>

volatile int flag = 0;

int main() {

while (flag == 0);

printf("Flag changed!\n");

return 0;

}

* 1. Sizeof
* sizeof is used to determine the size (in bytes) of a data type or variable at compile-time. It is a compile-time operator, not a function
* usage:

#include <stdio.h>

int main() {

int x = 10;

printf("Size of int: %zu bytes\n", sizeof(int));

printf("Size of x: %zu bytes\n", sizeof(x));

return 0;

}

* 1. const
* const is used to define variables whose values cannot be changed once initialized. This is useful for defining constants or read-only variables.
* Usage:

#include <stdio.h>

int main() {

const int MAX = 100;

printf("%d\n", MAX);

// MAX = 200;

return 0;

}

1. Explain the difference between the following variables.
   1. char \*ptr = “ABC”;

* ptr points to a string literal stored in read-only memory.
* You **cannot manipulate** the contents of the string literal.
* "ABC" is a **string literal**.
  1. char arr[]=”ABC”;
* arr is a writable array of characters initialized with the string "ABC".
* You **can manipulate** the contents of arr because it's a character array, not a string literal.
* "ABC" is a string literal used to initialize the array, but arr itself is not a string literal.

Can you manipulate the contents of ptr? Why?

Can you manipulate the contents of arr? Why?

Which one of the above is a string literal?

* **char \*ptr = "ABC";** is a **string literal** because the string "ABC" is a literal and is stored in a read-only section of memory. The pointer ptr just points to this literal.

1. Predict the output of the following code .

void main()

{

//set a and b both equal to 5.

int a=5, b=5;

//Print them and decrementing each time.

//Use postfix mode for a and prefix mode for b.

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

printf("\n%d %d",b++,--b);

}

Ans) 5 4

4 4

5.Refer the code snippet. It fails with error. Fix it.

#include<stdio.h>

int main()

{

int i,k;

const int num;

/\* for(i = 0;i < 9;i++)

{

k = k + 1;

} \*/

num = num + k; /\* Compiler gives the error here \*/

printf("final value of k:%d\n",k);

printf("value of num:%d\n",num);

return 0;

}

Ans)

#include<stdio.h>

int main()

{

int i,k;

int num;

/\* for(i = 0;i < 9;i++)

{

k = k + 1;

} \*/

num = num + k; /\* Compiler gives the error here \*/

printf("final value of k:%d\n",k);

printf("value of num:%d\n",num);

return 0;

}

Herein line num=num+k is the error beacuse const keyword is used to give the values with constant so we cannot be change in the num=num+k; so remove const from the int num;

Output:

Final value of k:0

Value of num: 0  
6. Consider the following code snippet. Evaluate the value of f1, f2 and f3.

int main()

{

int i = 10;

int j = 3;

float f1 = i / j;

float f2 = (float ) i / j;

float f3 = (float ) (i / j);

}

Ans)

Value of f1: 3.000000

Value of f2: 3.333333

Value of f3: 3.000000