# 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 - invalid
  2. int \_numvalue - valid
  3. float price\_money - valid
  4. char name1234567890123456789012345678901234567890 -valid
  5. char name value -invalid
  6. char $name - invalid

1. What is the meaning of the following keywords, show the usage
   1. Auto - In C, the auto keyword is used to declare local variables with automatic storage duration. This is the default for local variables, so you generally do not need to use auto explicitly. However, it's technically available

auto int x = 10;

* 1. Extern - The extern keyword is used to declare a variable or function that is defined in another file or in another part of the program. It indicates that the variable or function has external linkage.

int x = 10;

extern int x;

void printX() {

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

}

* 1. Volatile - The volatile keyword is used to indicate that a variable's value may change at any time, without any action being taken by the code the compiler sees. This is commonly used for variables that can be changed by external factors, such as hardware, or in multi-threaded environments where one thread modifies a variable and another thread reads it.

volatile int flag;

void waitForFlag() {

while (flag == 0) {

}

}

* 1. Sizeof - The sizeof operator in C is used to determine the size, in bytes, of a data type or variable. It is a compile-time operator

int a = 10;

double b = 3.14;

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

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

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

* 1. Const - The const keyword is used to define variables whose value cannot be changed after initialization. It enforces immutability and is often used to protect data that should not be modified.

const int max\_value = 100;

void function() {

const int\* ptr = &max\_value;

}

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

Can you manipulate the contents of ptr? Why?

Ans: no we can’t manipulate the contents of ptr because it is read only here.

Can you manipulate the contents of arr? Why?

Ans: yes we can manipulate the contents of ptr because it is write and can be modified here

Which one of the above is a string literal?

Ans: both are string literals and the string literals is”ABC”

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);

}

Output:

5 4

4 4

1. 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;

}

output:

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;

}

1. 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);

}

Output:

F1 : 3.000000

F2: 3.333333

F3: 3.000000