Keywords in C

In C Programming language, there are many rules so to avoid different types of errors. One of such rules is not able to declare variable names with auto, long, etc. This is all because these are keywords. Let us check all keywords in C language.

What are Keywords?

Keywords are predefined or reserved words that have special meanings to the compiler. These are part of the syntax and cannot be used as identifiers in the program. A list of keywords in C or reserved words in the C programming language are mentioned below:

auto	break	case	char	const	continue	default	do
double	else	enum	extern	float	for	goto	if
int	long	register	return	short	signed	sizeof	static
struct	switch	typedef	union	unsigned	void	volatile	while

- **auto**

auto is the default storage class variable that is declared inside a function or a block. auto variables can only be accessed within the function/block they are declared. By default, auto variables have garbage values assigned to them. Automatic variables are also called local variables as they are local to a function.

auto int num;

Here num is the variable of the storage class auto and its type is int. Below is the C program to demonstrate the auto keyword:

- C

```
// C program to demonstrate
// auto keyword
#include <stdio.h>

int printvalue()
{
  auto int a = 10;
  printf("%d", a);
}

// Driver code
int main()
{
  printvalue();
  return 0;
}
```

Output

```
10
```

❖ **break and continue**

The break statement is used to terminate the innermost loop. It generally terminates a loop or a switch statement. The switch statement skips to the next iteration of the loop. Below is the C program to demonstrate break and continue in C:

- C

```c
// C program to show use

// of break and continue

#include <stdio.h>


// Driver code

int main()

{

  for (int i = 1; i <= 10; i++)

  {

    if (i == 2)

    {

      continue;

    }

    if (i == 6)

    {

      break;

    }

    printf("%d ", i);

  }
```

```
 return 0;

}
```

Output

```
1 3 4 5
```

switch, case, and default

The switch statement in C is used as an alternate to the if-else ladder statement. For a single variable i.e, switch variable it allows us to execute multiple operations for different possible values of a single variable.

```
switch(Expression)
{
   case '1': // operation 1
         break;
   case:'2': // operation 2
         break;
   default: // default statement to be executed
}
```

Below is the C program to demonstrate the switch case statement:

- C

```
// C program to demonstrate

// switch case statement

#include <stdio.h>



// Driver code

int main() {

 int i = 4;
 switch (i) {
   case 1:
     printf("Case 1\n");break;
   case 2:
     printf("Case 2\n");break;
   case 3:
     printf("Case 3\n");break;
   case 4:
     printf("Case 4\n");break;
   default:
     printf("Default\n");break;
 }
}
```

Output

```
Case 4
```

Note: it is best to add a break statement after every case so that switch statement doesn't continue checking the remaining cases.

Output

```
Case 4

Default
```

char

char keyword in C is used to declare a character variable in the C programming language.

char x = 'D';

Below is the C program to demonstrate the char keyword:

- C

```
// C program to demonstrate

// char keyword

#include <stdio.h>



// Driver code

int main() {

  char c = 'a';

  printf("%c", c);

  return 0;

}
```

Output

```
a
```

const

The const keyword defines a variable who’s value cannot be changed.

const int num = 10;

Below is the C program to demonstrate the const keyword:

- C

```
// C program to demonstrate

// const keyword

#include <stdio.h>



// Driver code
```

```c
int main() {
  const int a = 11;
  a = a + 2;
  printf("%d", a);
  return 0;
}
```

This code will produce an error because the integer a was defined as a constant and it’s value was later on changed.

Output:

```
error: assignment of read-only variable 'a'
       a = a + 2;
```

do

The do statement is used to declare a do-while loop. A do-while loop is a loop that executes once, and then checks it’s condition to see if it should continue through the loop. After the first iteration, it will continue to execute the code while the condition is true.

Below is the C program to demonstrate a do-while loop.

- C

```c
// C program to demonstrate
// do-while keyword
#include <stdio.h>

// Driver code
int main()
{
  int i = 1;
  do {

    printf("%d ", i);

    i++;

  } while(i <= 5);


  return 0;

}
```

Output

```
1 2 3 4 5
```

double and float

The doubles and floats are datatypes used to declare decimal type variables. They are similar, but doubles have 15 decimal digits, and floats only have 7.

Example:

float marks = 97.5;

double num;

Below is the C program to demonstrate double float keyword:

- C

```
// C program to demonstrate

// double float keyword

#include <stdio.h>


// Driver code

int main() {

  float f = 0.3;
  double d = 10.67;

  printf("Float value: %f\n", f);

  printf("Double value: %f\n", d);

  return 0;

}
```

Output

```
Float value: 0.300000

Double value: 10.670000
```

if-else

The if-else statement is used to make decisions, where if a condition is true, then it will execute a block of code; if it isn't true (else), then it will execute a different block of code.

```
if(marks == 97) {
  // if marks are 97 then will execute this block of code
}
else {
  // else it will execute this block of code
}
```

Below is the C program to demonstrate an if-else statement:

- C

```
// C program to demonstrate

// if-else keyword

#include <stdio.h>



// Driver code
```

```
int main()

{

 int a = 10;

 if(a < 11)

 {

  printf("A is less than 11");

 }

 else

 {

  printf("A is equal to or "

         "greater than 11");

 }

 return 0;

}
```

Output

```
A is less than 11
```

enum

The enum keyword is used to declare an enum (short for enumeration). An enum is a user-defined datatype, which holds a list of user-defined integer constants. By default, the value of each constant is it’s index (starting at zero), though this can be changed. You can declare an object of an enum and can set it’s value to one of the constants you declared before. Here is an example of how an enum might be used:

- C

```
// An example program to
```

```c
// demonstrate working of

// enum in C

#include<stdio.h>



// enum declaration:

enum week{Mon, Tue, Wed, Thur, Fri, Sat, Sun};



// Driver code

int main()

{

//object of the enum (week), called day

  enum week day;

  day = Wed;

  printf("%d", day);

  return 0;

}
```

Output

```
2
```

extern

The extern keyword is used to declare a variable or a function that has an external linkage outside of the file declaration.

- C

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

```
extern int a;

int main(){

   printf("%d", a);

     return 0;
}
```

for

The “for” keyword is used to declare a for-loop. A for-loop is a loop that is specified to run a certain amount of times.

Below is the C program to demonstrate a for-loop:

- C

```
// C program to demonstrate
// for keyword
#include <stdio.h>

// Driver code
int main()
{
  for (int i = 0; i < 5; i++)
  {
    printf("%d ", i);

  }

  return 0;

}
```

Output

```
0 1 2 3 4
```

goto

The goto statement is used to transfer the control of the program to the given label. It is used to jump from anywhere to anywhere within a function.

Example:

```
goto label;

// code

label:
```

Below is the C program to demonstrate the goto keyword:

- C

```
// C program demonstrate

// goto keyword

#include <stdio.h>



// Function to print numbers

// from 1 to 10

void printNumbers() {

    int n = 1;
label:
    printf("%d ", n);
    n++;
    if (n <= 10) goto label;
}

// Driver code
int main(){
    printNumbers();
    return 0;
}
```

Output

```
1 2 3 4 5 6 7 8 9 10
```

int

int keyword is used in a type declaration to give a variable an integer type. In C, the integer variable must have a range of at least -32768 to +32767.

Example:

int x = 10;

Below is the C program to show the int keyword:

- C

```
// C program to demonstrate
// int keyword
#include <stdio.h>
```

```
void sum() {
    int a = 10, b = 20;
    int sum;
    sum = a + b;
    printf("%d", sum);
}

// Driver code
int main() {
    sum();
    return 0;
}
```

Output

```
30
```

short, long, signed, and unsigned

Different data types also have different ranges up to which they can store numbers. These ranges may vary from compiler to compiler. Below is a list of ranges along with the memory requirement and format specifiers on the **32-bit GCC compiler**.

Data Type	**Memory (bytes)**	**Range**	**Format Specifier**
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu

Data Type	**Memory (bytes)**	**Range**	**Format Specifier**
unsigned int	4	0 to 4,294,967,295	%u
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2^63) to (2^63)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
long double	16	3.4E-4932 to 1.1E+4932	%Lf

Below is the C program to demonstrate the short, long, signed, and unsigned keywords:

- C

```
// C program to demonstrate
// short, long, signed,
// and unsigned keyword
#include <stdio.h>


// Driver code
```

```
int main() {
  // short integer
  short int a = 12345;


  // signed integer
  signed int b = -34;


  // unsigned integer
  unsigned int c = 12;


  // L or l is used for
  // long int in C.
  long int d = 99998L;


  printf("Integer value with a short int data: %hd", a);
  printf("\nInteger value with a signed int data: %d", b);
  printf("\nInteger value with an unsigned int data: %u", c);
  printf("\nInteger value with a long int data: %ld", d);
  return 0;
}
```

Output

```
Integer value with a short int data: 12345

Integer value with a signed int data: -34
```

Integer value with an unsigned int data: 12

Integer value with a long int data: 99998

return

The return statement returns a value to where the function was called.

Example:

return x;

Below is the C program to demonstrate the return keyword:

- C

```
// C program to demonstrate

// return keyword

#include <stdio.h>

int sum(int x, int y) {

  int sum;

  sum = x + y;

  return sum;

}



// Driver code

int main() {

  int num1 = 10;

  int num2 = 20;

  printf("Sum: %d",

         sum(num1, num2));

  return 0;
}
```

Output

```
Sum: 30
```

sizeof

sizeof is a keyword that gets the size of an expression, (variables, arrays, pointers, etc.) in bytes.

Example:

sizeof(char);

sizeof(int);

sizeof(float); in bytes.

Below is the C program to demonstrate sizeof keyword:

- C

```
// C program to demonsstrate
// sizeof keyword
#include <stdio.h>

// Driver code
int main() {
  int x = 10;
  printf("%d", sizeof(x));
  return 0;
}
```

Output

```
4
```

register

Register variables tell the compiler to store variables in the CPU register instead of memory. Frequently used variables are kept in the CPU registers for faster access.

Example:

register char c = 's';

static

The static keyword is used to create static variables. A static variable is not limited by a scope and can be used throughout the program. It's value is preserved even after it's scope.

For Example:

static int num;

struct

The struct keyword in C programming language is used to declare a structure. A structure is a list of variables, (they can be of different data types), which are grouped together under one data type.

For Example:

```
struct Geek {
    char name[50];
    int num;
    double var;
};
```

Below is the C program for the struct keyword:

- C

```
// C program to demonstrate

// struct keyword

#include <stdio.h>

#include <string.h>


struct Books {

  char  title[50];
  char  author[50];
};

// Driver code
int main( ) {
  // Declare Book1 of type Book
  struct Books book1;

 // book 1 specification
 strcpy(book1.title, "C++ Programming");
 strcpy(book1.author, "Bjarne Stroustrup");

 // Print book details
 printf("Book 1 title : %s\n", book1.title);
 printf("Book 1 author : %s\n", book1.author);
 return 0;
}
```

Output

```
Book 1 title : C++ Programming

Book 1 author : Bjarne Stroustrup
```

typedef

The typedef keyword in C programming language is used to define a data type with a new name in the program. typedef keyword is used to make our code more readable.

For Example:

typedef long num

In this example we have changed the datatype name of “long” to “num”.

union

The union is a user-defined data type. All data members which are declared under the union keyword share the same memory location.

Example:

union GeekforGeeks {
 int x;
 char s;
} obj;

Below is the C program for the union keyword:

- C

```
#include <stdio.h>

union student {

  int age;

  char marks;

} s;



// Driver code

int main() {

  s.age = 15;

  s.marks = 56;

  printf("age = %d", s.age);

  printf("\nmarks = %d", s.marks);

}
```

Output

```
age = 56

marks = 56
```

void

The void keyword means nothing i.e, NULL value. When the function return type is used as the void, the keyword void specifies that it has no return value.

Example:

```
void fun() {
   // program
}
```

volatile

The volatile keyword is used to create volatile objects. Objects which are declared volatile are omitted from optimization as their values can be changed by code outside the scope of the current code at any point in time.

For Example:

```
const volatile marks = 98;
```

marks are declared constant so they can’t be changed by the program. But hardware can change it as they are volatile objects.

Conclusion

In this article, the points we learned about the keywords are mentioned below:

- Keywords are Reserved words in C with certain meanings.
- We can’t use keywords as any element’s name.
- There are 32 keywords in C all having unique meanings.