<http://www.programiz.com/c-programming/list-all-keywords-c-language>

| Keywords in C Programming |
| --- |
| [auto](http://www.programiz.com/c-programming/list-all-keywords-c-language#auto) |
| [break](http://www.programiz.com/c-programming/list-all-keywords-c-language#break_continue) |
| [case](http://www.programiz.com/c-programming/list-all-keywords-c-language#switch_case_default) |
| [char](http://www.programiz.com/c-programming/list-all-keywords-c-language#char) |
| [const](http://www.programiz.com/c-programming/list-all-keywords-c-language#const) |
| [continue](http://www.programiz.com/c-programming/list-all-keywords-c-language#break_continue) |
| [default](http://www.programiz.com/c-programming/list-all-keywords-c-language#switch_case_default) |
| [do](http://www.programiz.com/c-programming/list-all-keywords-c-language#do_while) |
| [double](http://www.programiz.com/c-programming/list-all-keywords-c-language#double_float) |
| [else](http://www.programiz.com/c-programming/list-all-keywords-c-language#if_else) |
| [enum](http://www.programiz.com/c-programming/list-all-keywords-c-language#enum) |
| [extern](http://www.programiz.com/c-programming/list-all-keywords-c-language#extern) |
| [float](http://www.programiz.com/c-programming/list-all-keywords-c-language#double_float) |
| [for](http://www.programiz.com/c-programming/list-all-keywords-c-language#for) |
| [goto](http://www.programiz.com/c-programming/list-all-keywords-c-language#goto) |
| [if](http://www.programiz.com/c-programming/list-all-keywords-c-language#if_else) |
| [int](http://www.programiz.com/c-programming/list-all-keywords-c-language#int) |
| [long](http://www.programiz.com/c-programming/list-all-keywords-c-language#short_long_signed_unsigned) |
| [register](http://www.programiz.com/c-programming/list-all-keywords-c-language#register) |
| [return](http://www.programiz.com/c-programming/list-all-keywords-c-language#return) |
| [short](http://www.programiz.com/c-programming/list-all-keywords-c-language#short_long_signed_unsigned) |
| [signed](http://www.programiz.com/c-programming/list-all-keywords-c-language#short_long_signed_unsigned) |
| [sizeof](http://www.programiz.com/c-programming/list-all-keywords-c-language#sizeof) |
| [static](http://www.programiz.com/c-programming/list-all-keywords-c-language#static) |
| [struct](http://www.programiz.com/c-programming/list-all-keywords-c-language#struct) |
| [switch](http://www.programiz.com/c-programming/list-all-keywords-c-language#switch_case_default) |
| [typedef](http://www.programiz.com/c-programming/list-all-keywords-c-language#typedef) |
| [union](http://www.programiz.com/c-programming/list-all-keywords-c-language#union) |
| [unsigned](http://www.programiz.com/c-programming/list-all-keywords-c-language#short_long_signed_unsigned) |
| [void](http://www.programiz.com/c-programming/list-all-keywords-c-language#void) |
| [volatile](http://www.programiz.com/c-programming/list-all-keywords-c-language#volatile) |
| [while](http://www.programiz.com/c-programming/list-all-keywords-c-language#do_while) |

## Description of all Keywords in C

**auto**

 auto  is used to define a variable of storage class automatic. For example:

auto int var1;

This statement suggests that var1 is a variable of storage class auto and data type int. Variables declared within function bodies are automatic by default. They are recreated each time a function is executed. Since, automatic variables are local to a function, automatic variables are also called local variables.

**break and continue**

The break statement is used to jump out of the innermost enclosing loop (while, do, for or switch statements) explicitly and pass control to next statement immediately following the loop. In other hand, continue statement is used to skip certain statements inside the loop.

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

if (i==3)

continue;

if (i==7)

break;

printf("%d ",i);

}

The output of the given code is:

1 2 4 5 6

It is because, when i=3, continue statement comes to effect and skips 3 and when i=7, break statements come to effect and terminates the loop.

**switch, case and default**

The switch statement tests the value of a expression and test with the different "case" values. We can use "default" value, if it doesn't matches any of "case" values. For example:

switch(option){

case '1':

//some statements to execute when 1

break;

case '5':

//some statements to execute when 5

break;

default:

//some statements to execute when default;

}

**char**

The char keyword is used for indicating the variable is of the type character. For example:

char variable0;

Here, variable0 is a variable of type character.

**const**

const makes the value of a pointer or a variable unmodifiable.

const int a=5;

In this example, a is the constant and its value is 5. This value cannot be changed in program.

**do and while**

while and do are used for looping in C. For example:

i=2;

while (i<10) {

print("%d ",i)

i++;

}

Output of the program

2 3 4 5 6 7 8 9.

If programmer finds easy to write statements to be executed before writing test condition, do...while loop is used.

int i;

do {

print("%d ",i);

i++;

}

while (i<10)

**double and float**

double and float are used for indicating floating type variables. Keywords float and double represents single precision and double precision floating point data respectively. For example:

float variable1;

double variable2;

Here, variable1 is single precision floating type variable whereas, variable2 is a double precision floating type variable.

**if and else**

if and else are used in decision making in C.

if (i==1)

printf("i is 1.")

else

prinf("i is not 1.")

If value of i is other than 1, output will be :

i is not 1

**enum**

enum is used to define enumerated type data type. Enumerated data type creates a set of constant of type int. For example:

enum enum\_var{

var1;

var2;

var3;

};

Here, a enumerated variable enum\_var is created having tags: var1, var2 and var3.

**extern**

Keyword extern is used for indicating the variable is external and is declared outside every function and can be accessed by any function. For example:

#include <stdio.h>

extern i=5;

void print1(){

printf ("%d ",i);

}

int main() {

printf("%d ",i);

}

return 0;

}

Output of the given code is:

5 5

**for**

Keyword for is used for looping in C. For example:

for (i=0;i<9;++i){

printf("%d ",i);

}

Output of the code given:

0 1 2 3 4 5 6 7 8

**goto**

Keyword goto is used for unconditional jump to a labeled statement inside that function. For example:

for(i=1;i<5;++i){

if (i==10)

goto error;

}

printf("i is not 10");

error:

printf("Error, count can't be 10");

Output of the given code:

**int**

int is used for indicating the variable is of type integer.

int var0;

Here, var0 is a variable of type integer.

**short, long, signed and unsigned**

short, long, signed and unsigned are type modifiers that alters the meaning of base data type to yield new type.

short int var1;

long int var2;

signed int var3;

unsigned int var4;

Here, the int type variable is modified to short int, long int, signed int and unsigned int respectively.

| Range of int type data types | |
| --- | --- |
| Datatypes | Range |
| short int | -32768 to 32767 |
| long int | -2147483648 to 214743648 |
| signed int | -32768 to 32767 |
| unsigned int | 0 to 65535 |

**return**

Keyword return terminates the execution of current function and returns the value to the calling function.

int func(){

int b=5;

return b;

}

This functionfunc() returns 5 to to the calling function.

**sizeof**

sizeof is used to find the number of bytes of an object.

#include <stdio.h>

int main(){

printf("%u bytes.",sizeof(char));

}

Output of the given code:

1 bytes.

Here, sizeof(...) is used to find the number of bytes of type char.

**register**

Variable of storage class register are much faster than normal variables.

register int var1;

Here, var1 is the variable of storage class register.

**static**

static is used for indicating the variable is of storage class static. The value of the static variables persists until the end of the program. For example:

static int var;

Here, var is a variable of storage class static.

**struct**

struct is used in creating a structure which provide means to group different types of variable under one name for easier handling.

struct student{

char name[80];

float marks;

int age;

}s1,s2;

Here, struct keyword is used in creating a structure of tag name student and variables of type "struct student".

**typedef**

Keyword typedef is used to explicitly associate a type with an identifier.

typedef float kg;

kg bear,tiger;

Here, the use of typedef is to create a type kg and this kg type is used in declaring variables bearand tiger.

**union**

Union is used in creating a union which provide means to group different types of variable under one name for easier handling.

union student {

char name[80];

float marks;

int age;

}

Here, union keyword is used in creating a union of tag name student and variables of type "union student".

**void**

void is used to indicate that a function takes no arguments or returns no value.

void no\_return(int a){

.....

}

Here, function no\_return( ) can take value but, can't return value because, the return type is void.

**volatile**

volatile is used to create a volatile object. A volatile object can be modified in unspecified way by the hardware.

const volatile number

Here, number is a volatile object. Since, number is constant variable, program can't change it but, hardware can change it because, it is a volatile object.