***32 Keyword’s in C***

**1) Auto:** The **auto** keyword is like asking C++ to figure out the type of a variable for you. Instead of writing the type yourself, the compiler guesses it based on the value you assign to the variable.

**Program (in C++):**

#include <iostream>

#include <typeinfo>

int main() {

int x = 10;

auto ptr = &x; // ptr is a pointer to x (int\*)

auto &ref = x; // ref is a reference to x (int&)

std::cout << ptr << std::endl;

std::cout << ref << std::endl;

std::cout << typeid(ref).name() << std::endl;

std::cout << typeid(ptr).name() << std::endl;

return 0;

}

**Program (in C):**

#include <stdio.h>

int main() {

auto int i = 10.1234;

printf("%d", i);

return 0;

}

**2) Break:** The **break** keyword is used:

* Inside loops: To exit a for, while, or do-while loop early.
* Inside switch: To exit a case in a switch statement after executing its code.

**Program:**

for (int i = 1; i <= 3; i++) {

for (int j = 1; j <= 3; j++) {

if (j == 2) break;

printf("i = %d, j = %d\n", i, j);

}

}

**3) Case:** The **case** keyword is part of the switch statement, enabling specific blocks of code execution based on a variable or expression.

**Program:**

#include <stdio.h>

int main() {

int num = 2;

switch (num) {

case 1:

printf("You chose One\n");

break;

case 2:

printf("You chose Two\n");

break;

case 3:

printf("You chose Three\n");

break;

default:

printf("Invalid choice\n");

}

return 0;

}

**4) Char:** The **char** is a fundamental data type used to store a single character. A char is typically 1 byte in size and can represent ASCII characters.

**Program:**

#include <stdio.h>

int main() {

char letter = 'A';

printf("Character: %c\n", letter);

return 0;

}

**5) Const:** The **const** keyword is used to declare variables whose values cannot be modified after initialization. It ensures that a variable is read-only.

**Program:**

#include <stdio.h>

int main() {

const int x = 10;

printf("Value of x: %d\n", x);

return 0;

}

**6) Continue:** The **continue** statement skips the current iteration of a loop and moves to the next iteration.

**Program:**

#include <stdio.h>

int main() {

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

if (i == 3) {

continue;

}

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

}

return 0;

}

**7) Default:** The **default** keyword in a switch statement defines a block of code that executes if no case labels match.

**Program:**

#include <stdio.h>

int main() {

int num = 5;

switch (num) {

case 1:

printf("Number is One\n");

break;

case 2:

printf("Number is Two\n");

break;

default:

printf("Number is not One or Two\n");

}

return 0;

}

**8) Do:** The **do...while** loop executes a block of code at least once, then repeats while the condition is true.

**Program:**

#include <stdio.h>

int main() {

int i = 1;

do {

printf("Value of i: %d\n", i);

i++;

} while (i <= 5);

return 0;

}

**9) Double:** The **double** keyword stores high-precision floating-point numbers.

**Program:**

#include <stdio.h>

int main() {

double pi = 3.141592653589793;

double radius, area;

printf("Enter the radius of the circle: ");

scanf("%lf", &radius);

area = pi \* radius \* radius;

printf("The area of the circle is: %.5f\n", area);

return 0;

}

**10) If:** The **if** keyword checks a condition. If true, the associated block of code is executed.

**Program:**

#include <stdio.h>

int main() {

int age = 18;

if (age >= 18) {

printf("You are an adult.\n");

}

return 0;

}

**11) Else:** The **else** keyword defines a block of code that executes when the if condition is false.

**Program:**

#include <stdio.h>

int main() {

int x = 5;

if (x > 10) {

printf("x is greater than 10\n");

} else {

printf("x is less than or equal to 10\n");

}

return 0;

}

**12) Enum:** The **enum** keyword defines a set of named integral constants.

**Program:**

#include <stdio.h>

enum Day { Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday };

int main() {

enum Day today = Wednesday;

printf("Today is day number %d\n", today);

return 0;

}

**13) Extern:** The **extern** keyword declares a variable defined elsewhere in the program.

**Program:**

#include <stdio.h>

extern int count;

int count = 10;

int main() {

printf("The count is: %d\n", count);

return 0;

}

**14) Float:** The **float** keyword is used for single-precision floating-point numbers.

**Program:**

#include <stdio.h>

int main() {

float temperature = 36.6;

printf("The temperature is: %.2f°C\n", temperature);

return 0;

}

**15) For:** The **for** keyword is used for loop iterations.

**Program:**

#include <stdio.h>

int main() {

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

printf("%d\n", 2 \* i);

}

return 0;

}

**16) Goto:** The **goto** keyword transfers control to another part of the program using a label.

**Program:**

#include <stdio.h>

int main() {

printf("Before the goto statement.\n");

goto end;

printf("This will be skipped.\n");

end:

printf("This is after the goto statement.\n");

return 0;

}

**17) Int:** The **int** keyword is used to declare integer variables to store whole numbers.

**Program:**

#include <stdio.h>

int main() {

int num = 42;

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

return 0;

}

**18) Long:** The **long** keyword is used to declare larger integer values compared to int.

**Program:**

#include <stdio.h>

int main() {

long num = 1234567890L;

printf("%ld\n", num);

return 0;

}

**19) Register:** The **register** keyword suggests storing a variable in a CPU register for faster access.

**Program:**

#include <stdio.h>

int main() {

register int i;

for (i = 0; i < 5; i++) {

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

}

return 0;

}

**20) Return:** The **return** keyword exits a function and optionally returns a value.

**Program:**

#include <stdio.h>

int sum(int a, int b) {

return a + b;

}

int main() {

int result = sum(3, 4);

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

return 0;

}

**21) Short:** The **short** keyword declares integer variables with a smaller range compared to int.

**Program:**

#include <stdio.h>

int main() {

short num = 32767;

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

return 0;

}

**22) Signed:** The **signed** keyword specifies that a variable can store both positive and negative values.

**Program:**

#include <stdio.h>

int main() {

signed int num = -10;

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

return 0;

}

**23) Sizeof:** The **sizeof** keyword determines the size of a data type or variable in bytes.

**Program:**

#include <stdio.h>

int main() {

int num = 10;

printf("%zu\n", sizeof(num));

return 0;

}

**24) Static:** The **static** keyword makes a variable retain its value between function calls.

**Program:**

#include <stdio.h>

void counter() {

static int count = 0;

count++;

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

}

int main() {

counter();

counter();

counter();

return 0;

}

**25) Struct:** The **struct** keyword groups multiple data types into a single unit.

**Program:**

#include <stdio.h>

struct Point {

int x;

int y;

};

int main() {

struct Point p1 = {10, 20};

printf("%d %d\n", p1.x, p1.y);

return 0;

}

**26) Switch:** The **switch** keyword executes a block of code based on a variable's value.

**Program:**

#include <stdio.h>

int main() {

int num = 2;

switch (num) {

case 1:

printf("One\n");

break;

case 2:

printf("Two\n");

break;

default:

printf("Other\n");

}

return 0;

}

**27) While:** The **while** loop executes a block of code as long as a condition is true.

**Program:**

#include <stdio.h>

int main() {

int count = 1;

while (count <= 5) {

printf("Count is: %d\n", count);

count++;

}

printf("Loop finished.\n");

return 0;

}

**28) Volatile:** The **volatile** keyword tells the compiler a variable's value may change unexpectedly.

**Program:**

#include <stdio.h>

volatile int flag = 0;

void interrupt\_service\_routine() {

flag = 1;

}

int main() {

while (flag == 0) {

// Do something

}

printf("Interrupt occurred.\n");

return 0;

}

**29) Void:** The **void** keyword is used for functions that do not return any value.

**Program:**

#include <stdio.h>

void printMessage() {

printf("Hello, World!\n");

}

int main() {

printMessage();

return 0;

}

**30) Unsigned:** The **unsigned** keyword specifies a variable can only store non-negative values.

**Program:**

#include <stdio.h>

int main() {

unsigned int num = 3000;

printf("The value of num is: %u\n", num);

return 0;

}

**31) Union:** The **union** keyword allows different data types to share the same memory location.

**Program:**

#include <stdio.h>

union Data {

int i;

float f;

char str[20];

};

int main() {

union Data data;

data.i = 10;

printf("data.i: %d\n", data.i);

data.f = 3.14;

printf("data.f: %.2f\n", data.f);

snprintf(data.str, sizeof(data.str), "Hello, Union!");

printf("data.str: %s\n", data.str);

return 0;

}

**32) Typedef:** The **typedef** keyword creates an alias for a data type.

**Program:**

#include <stdio.h>

typedef unsigned int uint;

int main() {

uint num = 10;

printf("The value of num is: %u\n", num);

return 0;

}