CS 332/532 Systems Programming

Lecture 3

- C Variables, Operators -

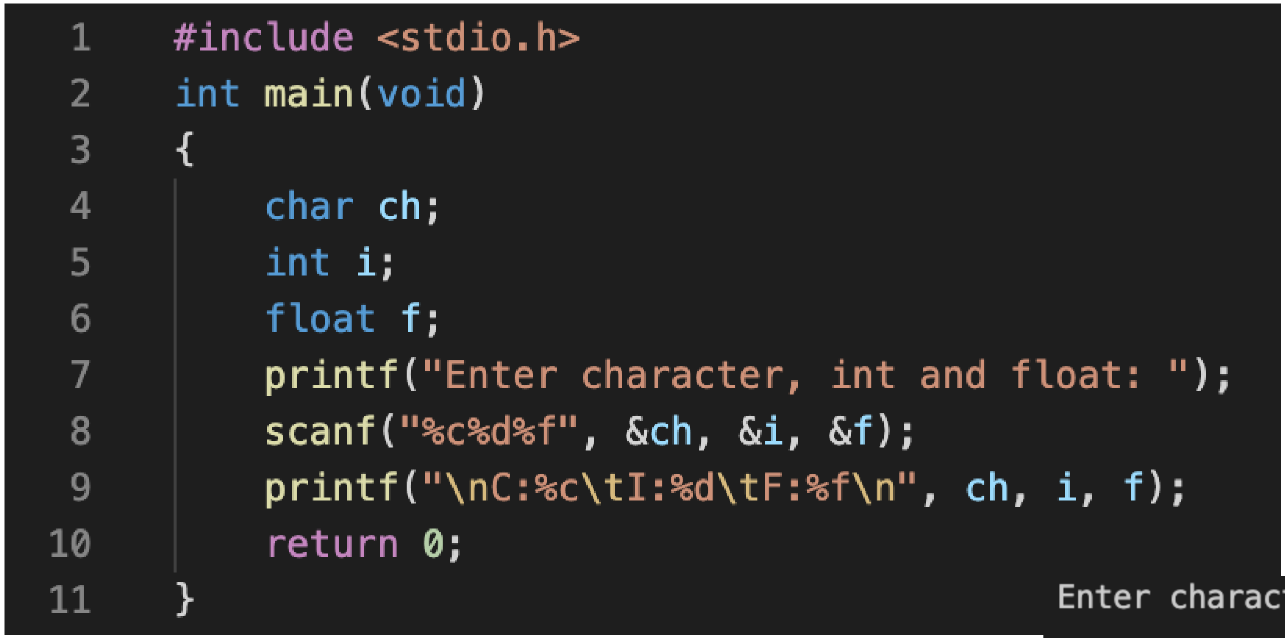
Professor : Mahmut Unan – UAB CS

Agenda

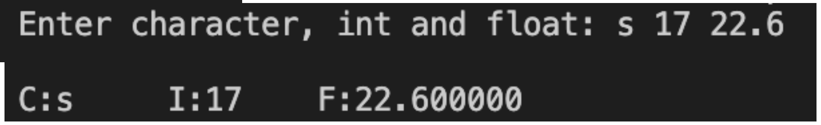
- Operators
- Branching
- Loops

scanf()

```
int i;
float j;
scanf("%d%f", &i, &j);

char str[100];
scanf("%s", str);
```



```
#include <stdio.h>
int main(void)
{
    char ch;
    int i;
    float f;
    printf("Enter character, int and float: ");
    scanf("%c%d%f", &ch, &i, &f);
    printf("\nC:%c\tI:%d\tF:%f\n", ch, i, f);
    return 0;
}
```



```
Enter character, int and float: s 17 22.6

C:s     I:17    F:22.600000
```

The assignment operator =

```
int a,b,c,d,e;
a=b=c=d=e=25;
```

or even the following is legal

```
a=25;
d=a + ( b = ( (e = a+10) + 40));
```

Arithmetic Operators

- + - / * %
- int/int = cuts off the decimal part

```
int a=7;
int b=5;
a/b will be equal to 1
```

also, be careful with the % operator

```
if ((n%2)==1) is dangerous**
if((n%2)!=0)  is safe
```

** if n is odd and negative

++ and --

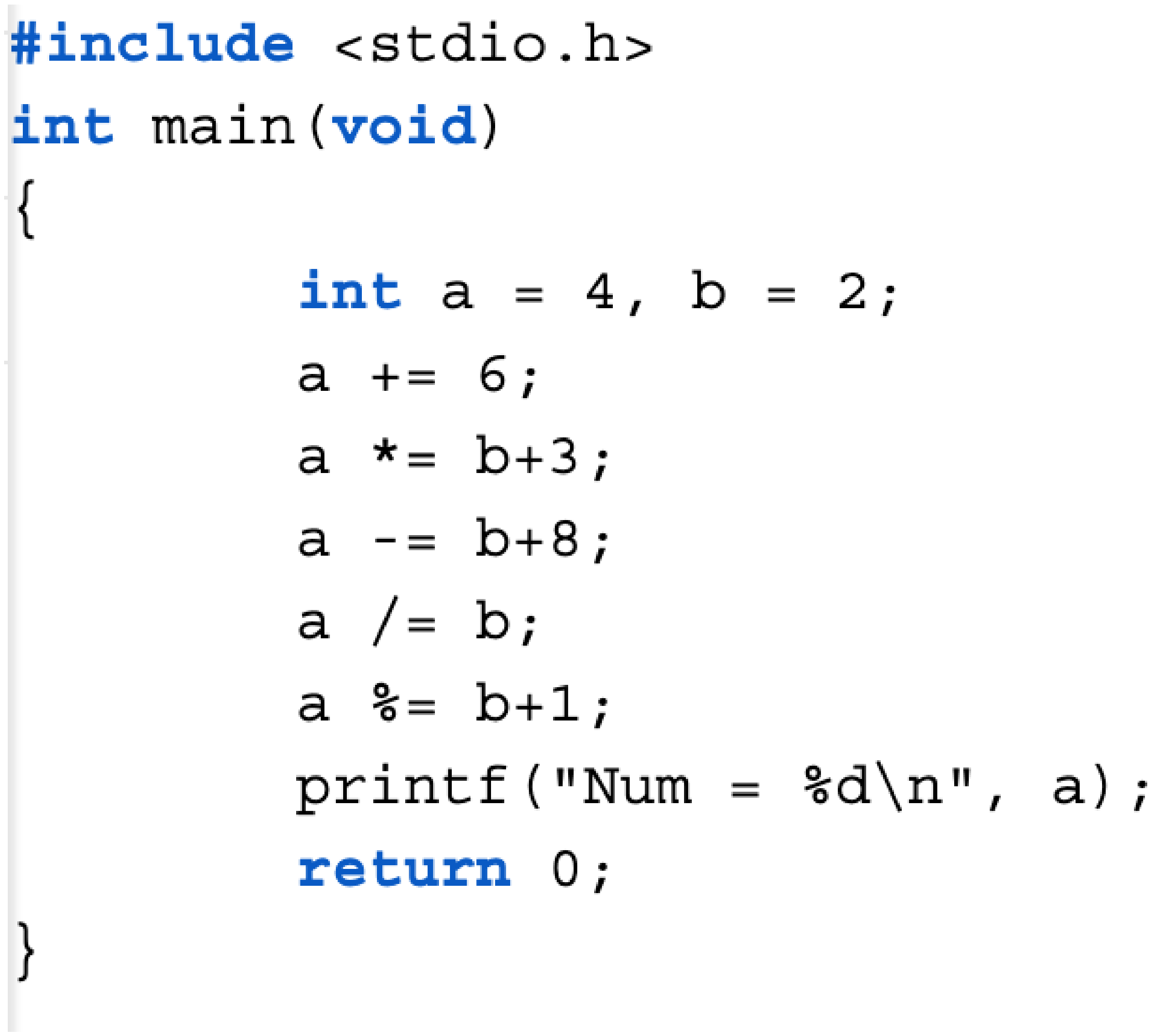
- Similar to Java

```
int a=25;
a++; /* is equal to a = a+1; */
a--; /* is equal to a = a-1; */

int c=5,d;
d=c++;
```

or

```
d=++c;
```

Compound Assignment Operators



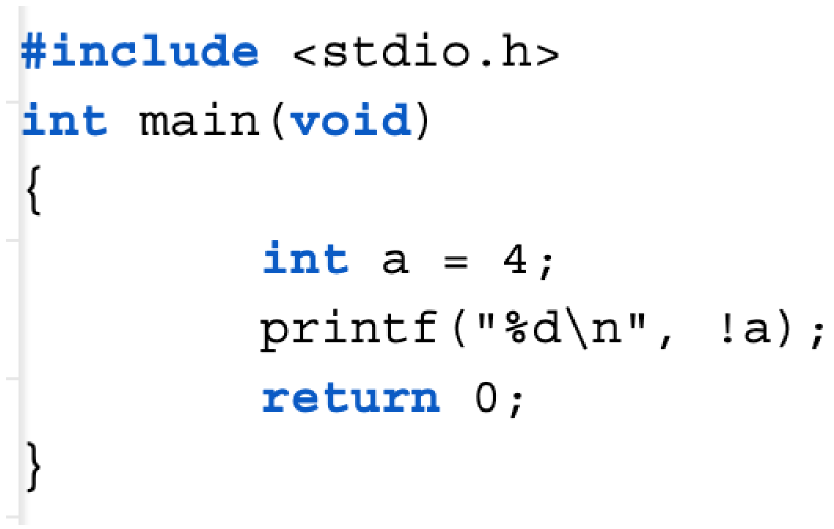
```c
#include <stdio.h>
int main(void)
{
        int a = 4, b = 2;
        a += 6;
        a *= b+3;
        a -= b+8;
        a /= b;
        a %= b+1;
        printf("Num = %d\n", a);
        return 0;
}
```

Comparisons

- >   >=   <   <=   !=   ==
- if (a == 10)

Logical Operators

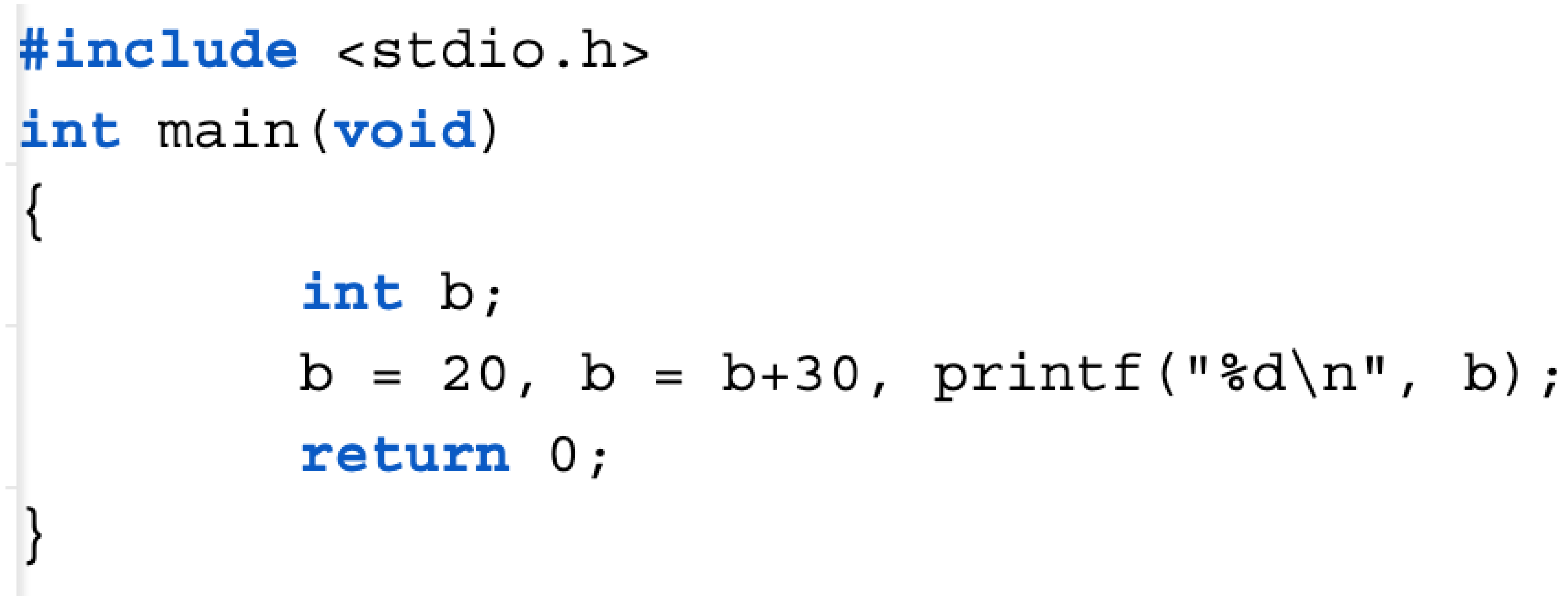
- ! not operator, && operator, || operator



```c
#include <stdio.h>
int main(void)
{
        int a = 4;
        printf("%d\n", !a);
        return 0;
}
```

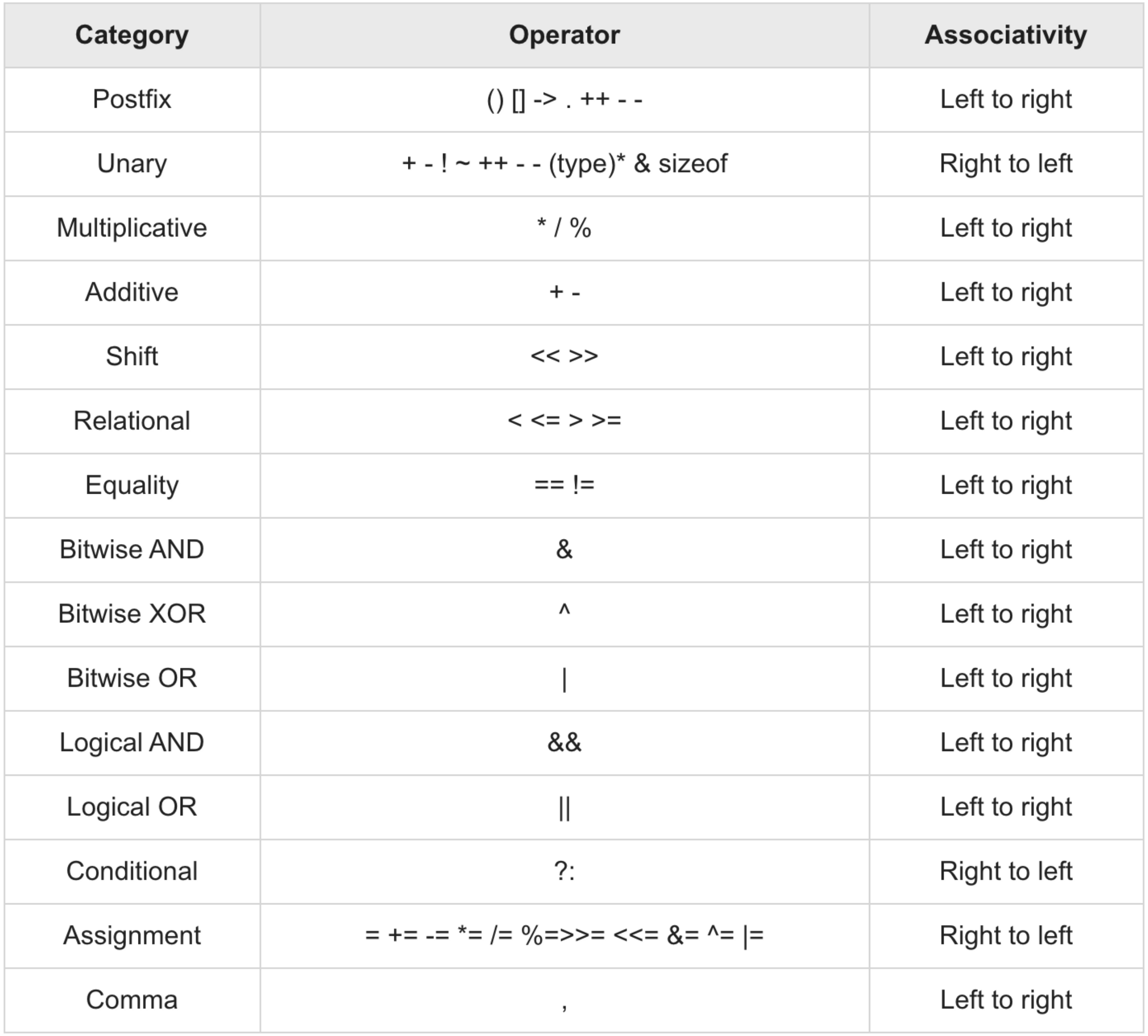
The Comma Operator

- The comma (,) operator can be used to merge several expressions to form a single expression



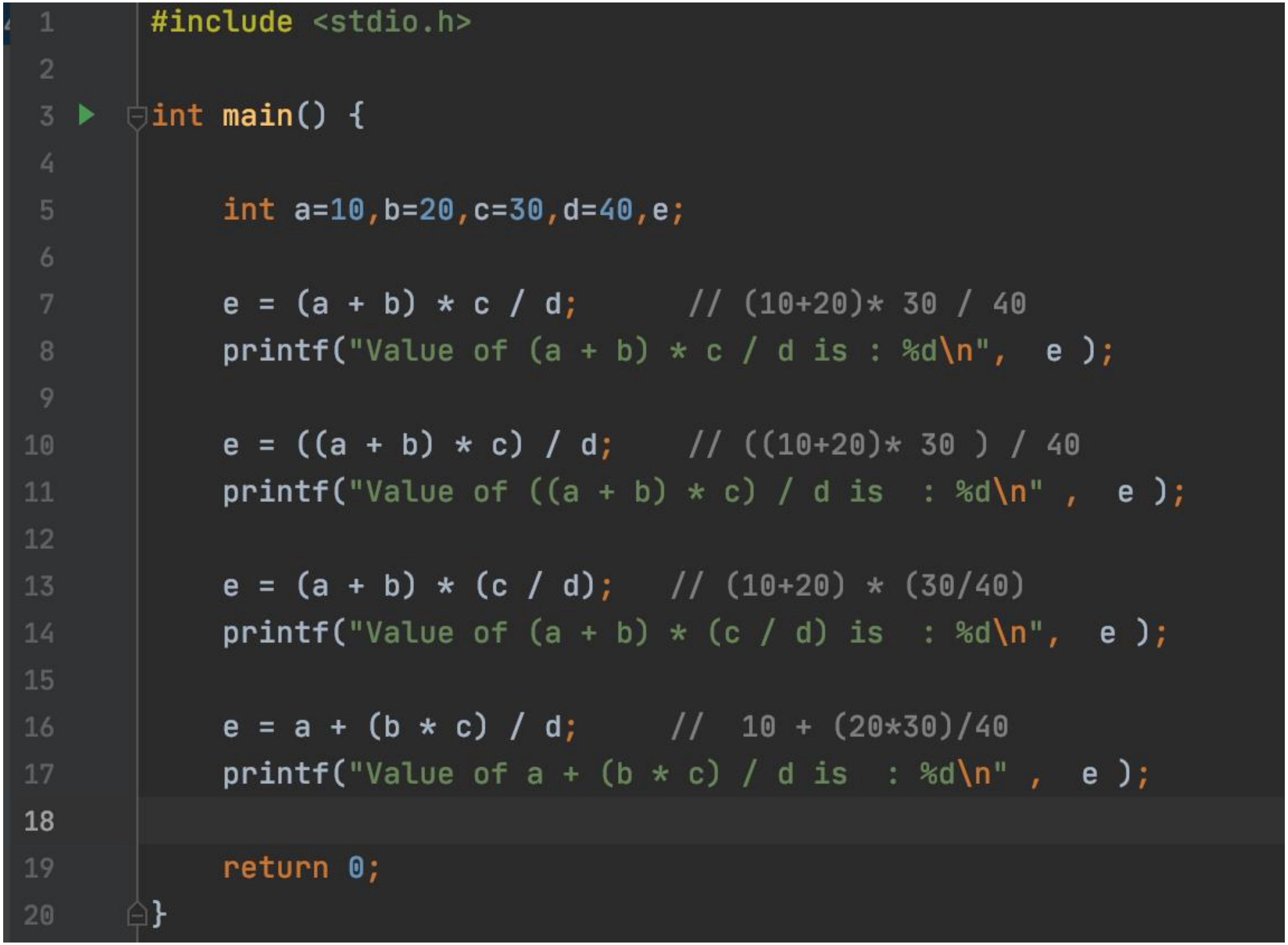
```c
#include <stdio.h>
int main(void)
{
        int b;
        b = 20, b = b+30, printf("%d\n", b);
        return 0;
}
```

Operator Precedence



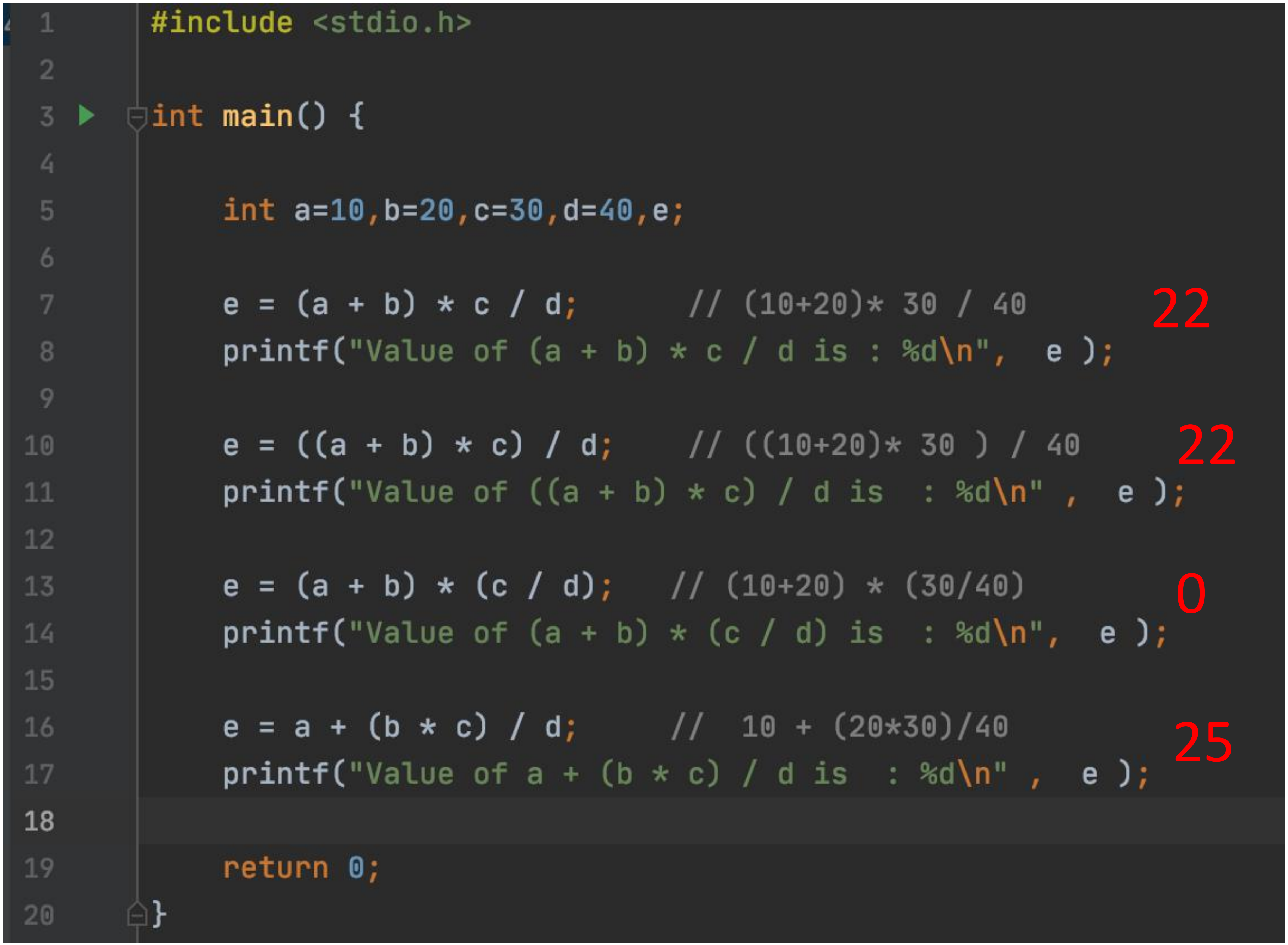
Category	Operator	Associativity
Postfix	() [] -> . ++ - -	Left to right
Unary	+ - ! ~ ++ - - (type)* & sizeof	Right to left
Multiplicative	* / %	Left to right
Additive	+ -	Left to right
Shift	<< >>	Left to right
Relational	< <= > >=	Left to right
Equality	== !=	Left to right
Bitwise AND	&	Left to right
Bitwise XOR	^	Left to right
Bitwise OR	|	Left to right
Logical AND	&&	Left to right
Logical OR	||	Left to right
Conditional	?:	Right to left
Assignment	= += -= *= /= %=>>= <<= &= ^= |=	Right to left
Comma	,	Left to right

https://www.tutorialspoint.com/cprogramming/c_operators_precedence.htm



```
#include <stdio.h>

int main() {

    int a=10,b=20,c=30,d=40,e;

    e = (a + b) * c / d;        // (10+20)* 30 / 40
    printf("Value of (a + b) * c / d is : %d\n",  e );

    e = ((a + b) * c) / d;      // ((10+20)* 30 ) / 40
    printf("Value of ((a + b) * c) / d is  : %d\n" ,  e );

    e = (a + b) * (c / d);    // (10+20) * (30/40)
    printf("Value of (a + b) * (c / d) is  : %d\n",  e );

    e = a + (b * c) / d;       //  10 + (20*30)/40
    printf("Value of a + (b * c) / d is  : %d\n" ,  e );

    return 0;
}
```

https://www.tutorialspoint.com/cprogramming/c_operators_precedence.htm



```c
#include <stdio.h>

int main() {

    int a=10,b=20,c=30,d=40,e;

    e = (a + b) * c / d;        // (10+20)* 30 / 40
    printf("Value of (a + b) * c / d is : %d\n",  e );

    e = ((a + b) * c) / d;      // ((10+20)* 30 ) / 40
    printf("Value of ((a + b) * c) / d is  : %d\n" ,  e );

    e = (a + b) * (c / d);    // (10+20) * (30/40)
    printf("Value of (a + b) * (c / d) is  : %d\n",  e );

    e = a + (b * c) / d;       //  10 + (20*30)/40
    printf("Value of a + (b * c) / d is  : %d\n" ,  e );

    return 0;
}
```

22

22

0

25

https://www.tutorialspoint.com/cprogramming/c_operators_precedence.htm

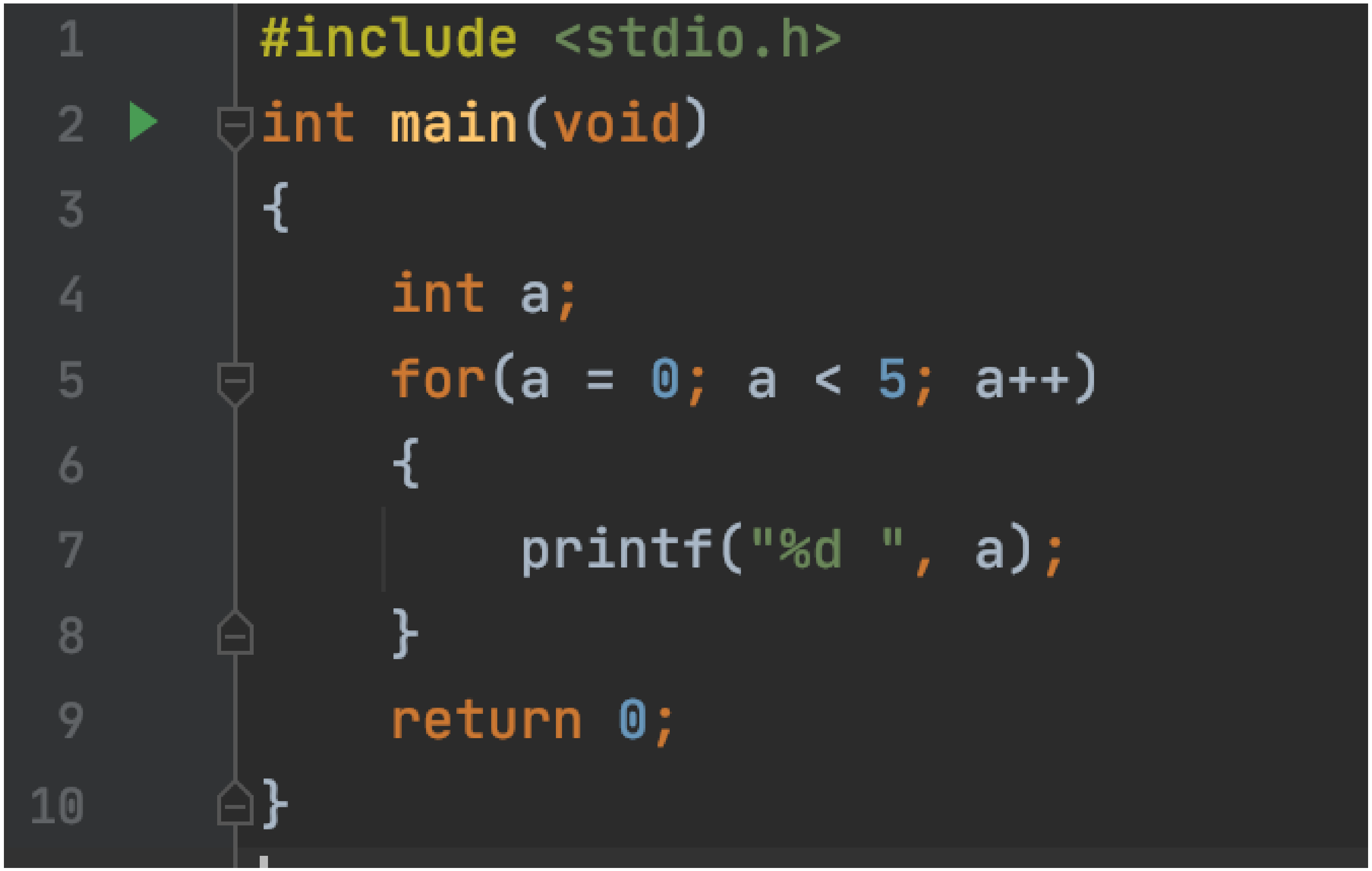
if else else if

```
#include <stdio.h>
int main(void)
{
        int a = 10, b = 20, c = 30;
        if(a > 5)
        {
                if(b == 20)
                        printf("1 ");
                if(c == 40)
                        printf("2 ");
                else
                        printf("3 ");
        }
        else
                printf("4\n");
        return 0;
}
```

switch statement

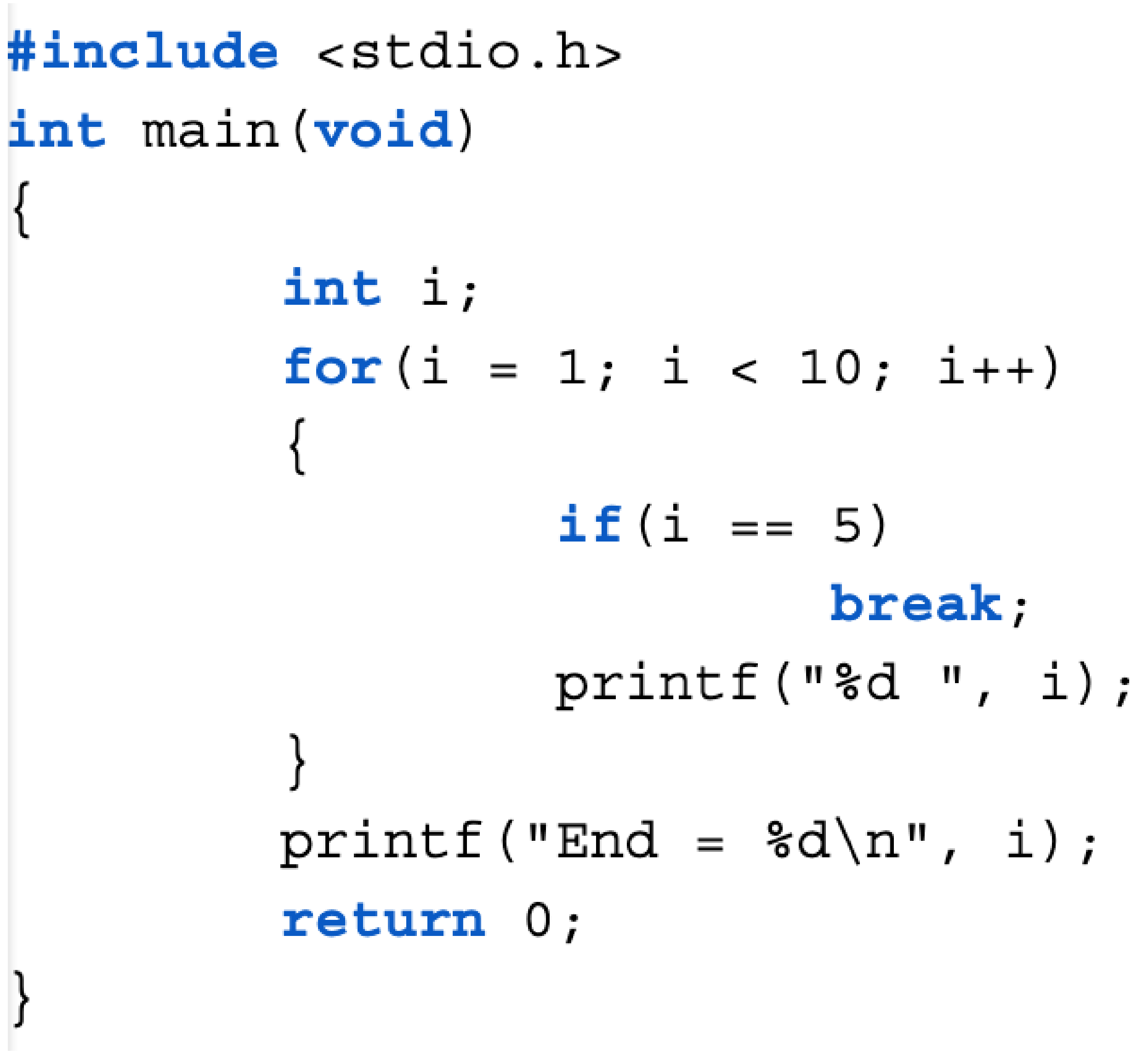
```c
#include <stdio.h>
int main(void)
{
        int a;
        printf("Enter number: ");
        scanf("%d", &a);
        switch(a)
        {
                case 1:
                        printf("One\n");
                break;
                case 2:
                        printf("Two\n");
                break;
                default:
                        printf("Other\n");
                break;
        }
        printf("End\n");
        return 0;
}
```

for loop



```c
#include <stdio.h>
int main(void)
{
    int a;
    for(a = 0; a < 5; a++)
    {
        printf("%d ", a);
    }
    return 0;
}
```

The `break` Statement

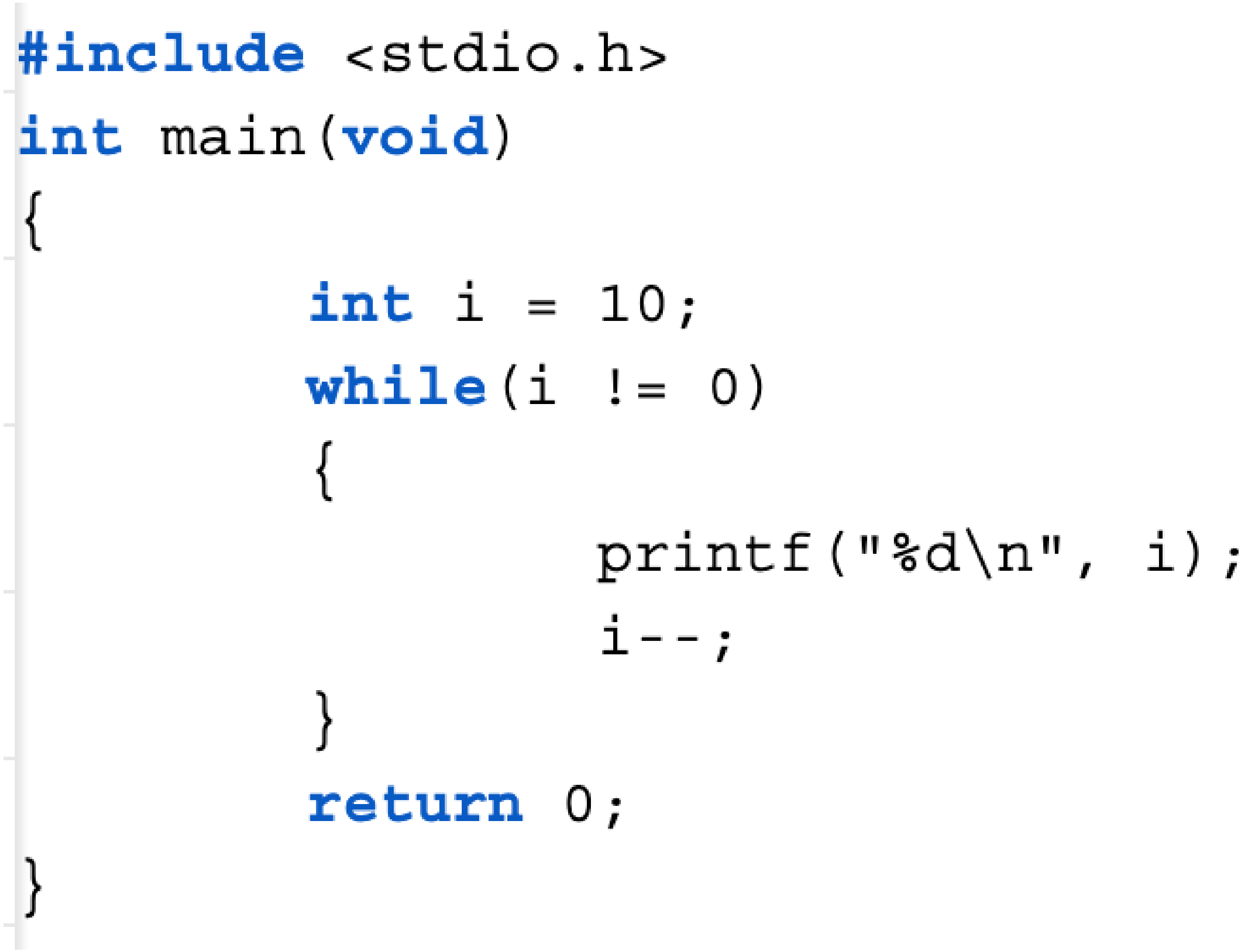


```c
#include <stdio.h>
int main(void)
{
        int i;
        for(i = 1; i < 10; i++)
        {
                if(i == 5)
                        break;
                printf("%d ", i);
        }
        printf("End = %d\n", i);
        return 0;
}
```

The continue Statement

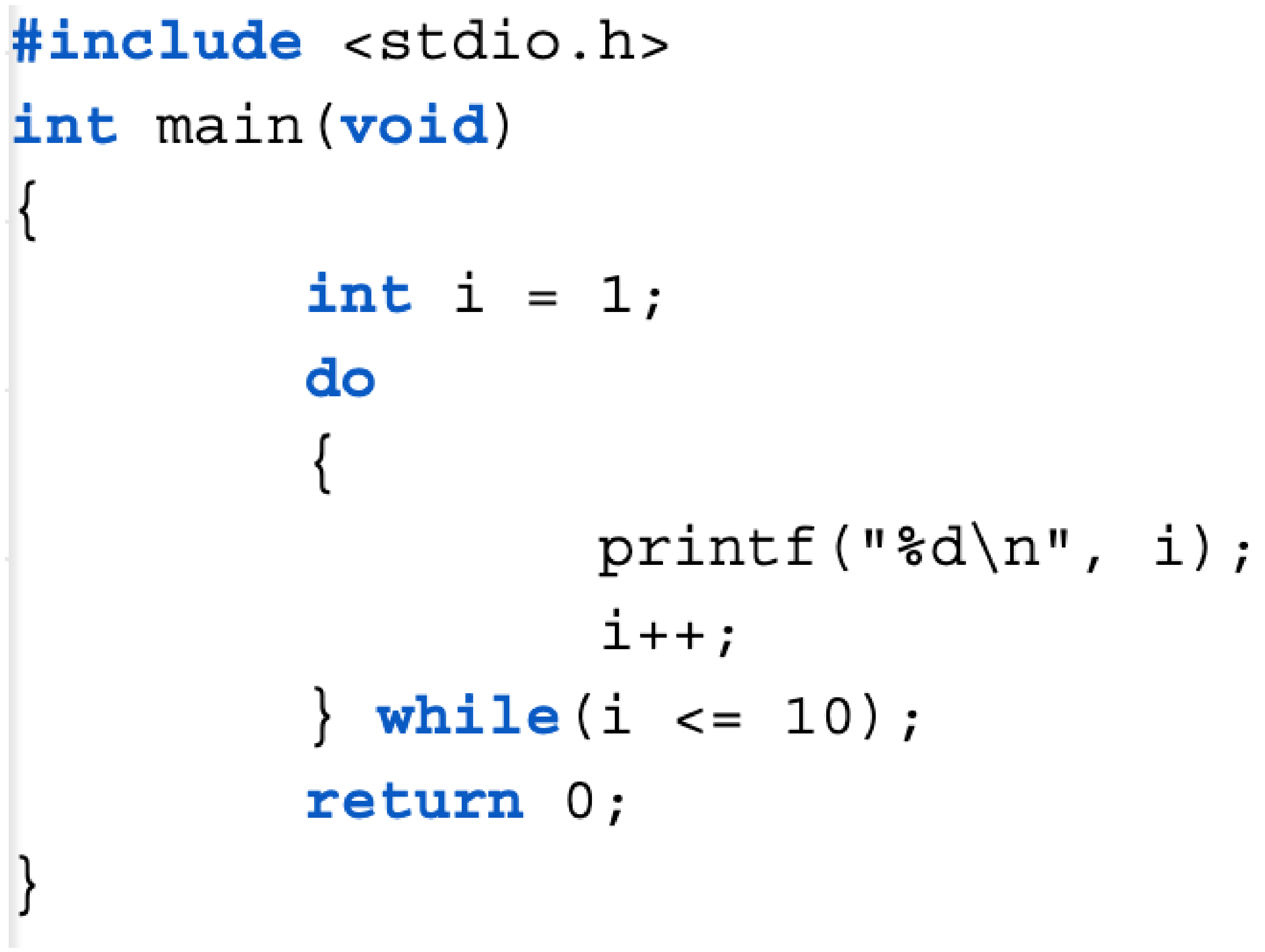
```
#include <stdio.h>
int main(void)
{
        int i;
        for(i = 1; i < 10; i++)
        {
                if(i < 5)
                        continue;
                printf("%d ", i);
        }
        return 0;
}
```

while loop

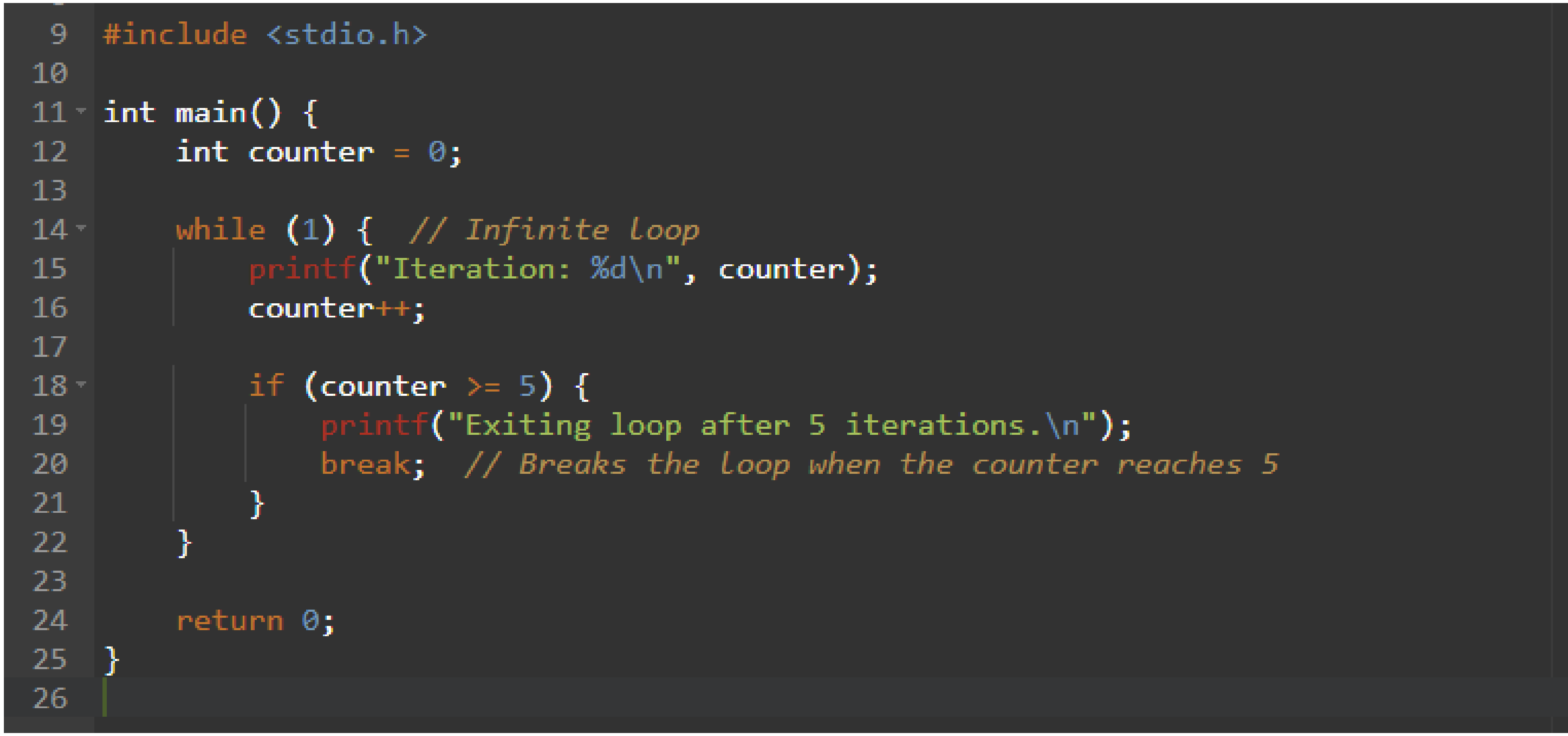


```c
#include <stdio.h>
int main(void)
{
        int i = 10;
        while(i != 0)
        {
                printf("%d\n", i);
                i--;
        }
        return 0;
}
```

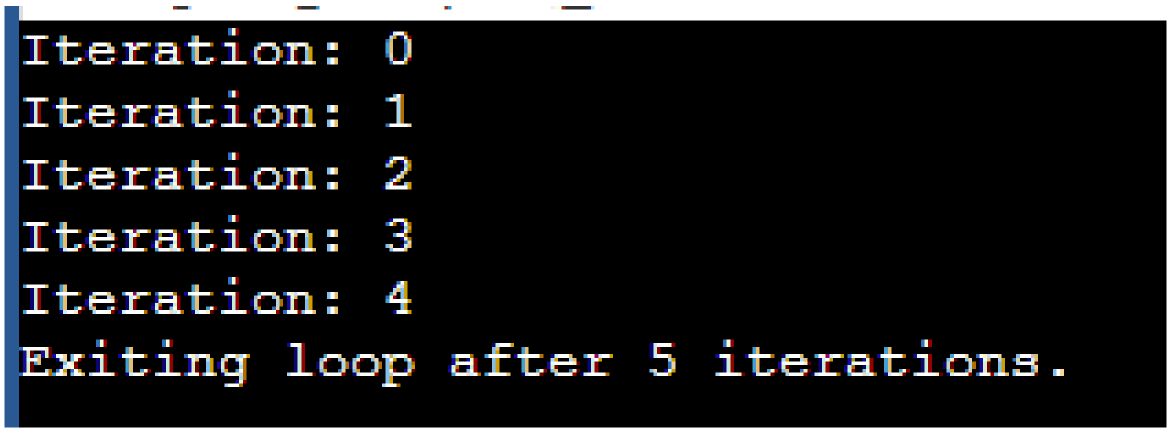
do-while loop



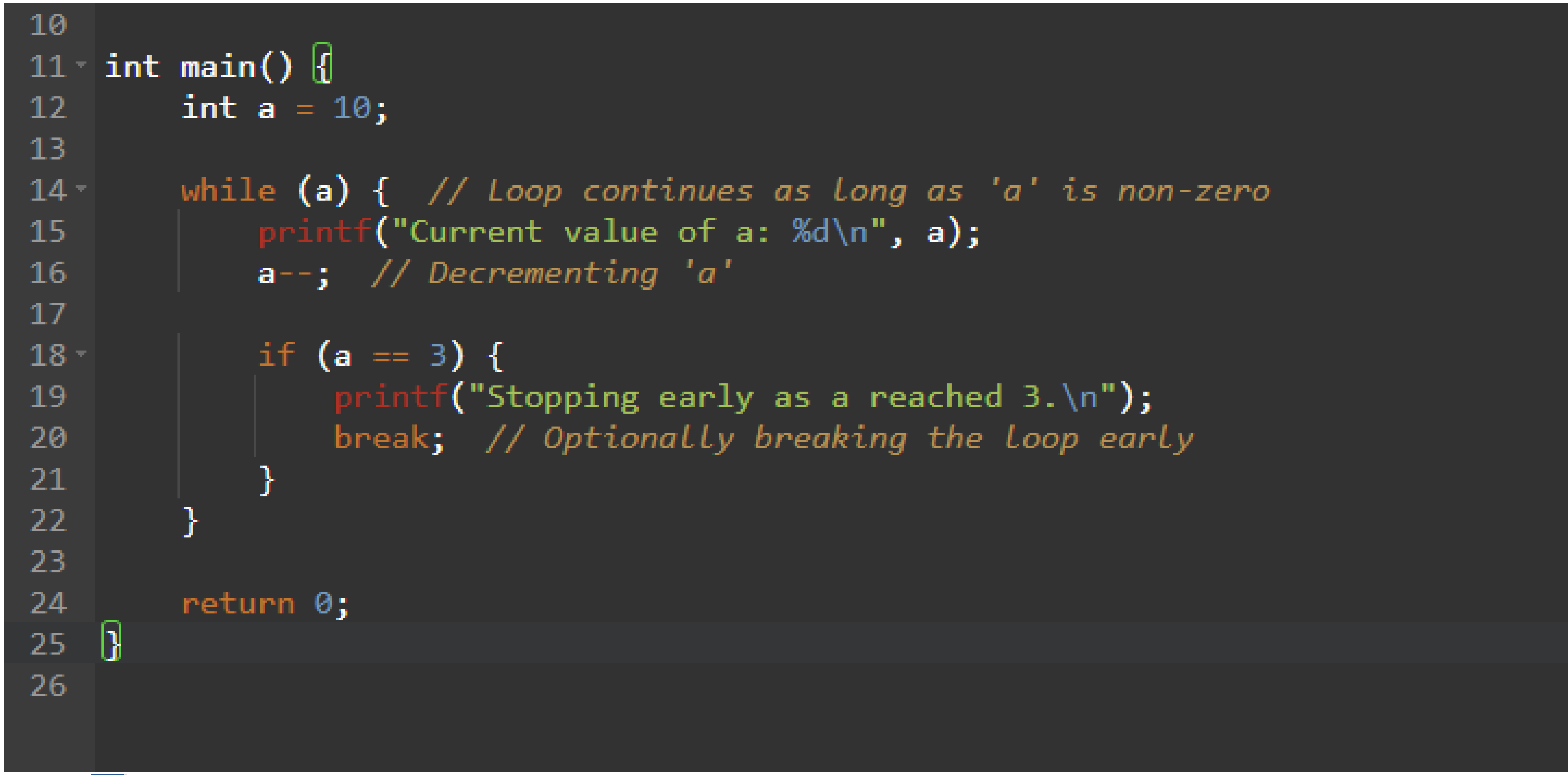
```c
#include <stdio.h>
int main(void)
{
        int i = 1;
        do
        {
                printf("%d\n", i);
                i++;
        } while(i <= 10);
        return 0;
}
```



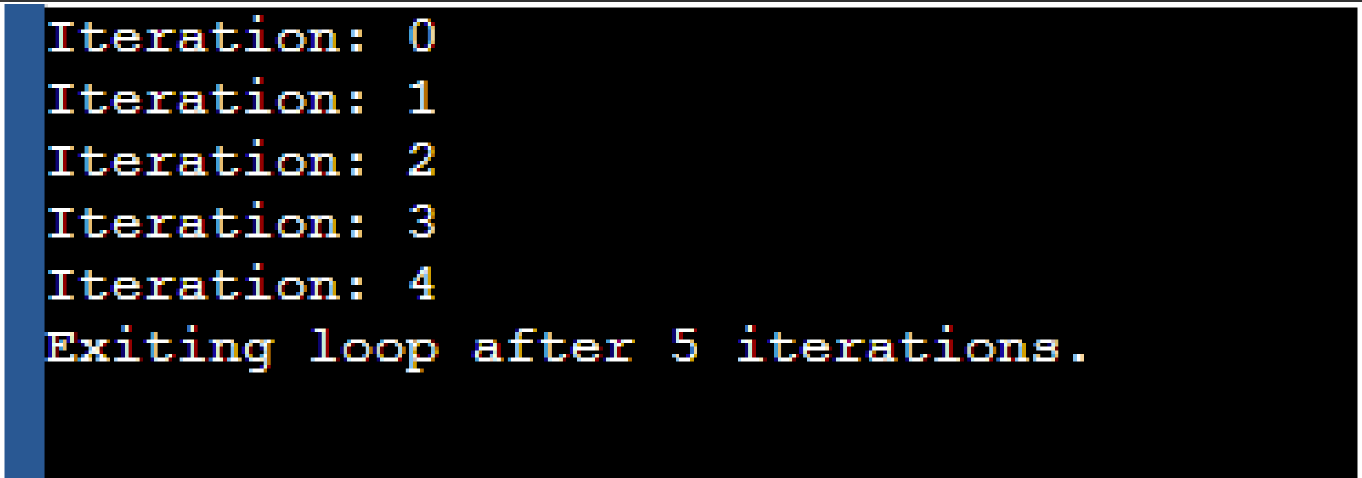
```c
#include <stdio.h>

int main() {
    int counter = 0;

    while (1) {  // Infinite loop
        printf("Iteration: %d\n", counter);
        counter++;

        if (counter >= 5) {
            printf("Exiting loop after 5 iterations.\n");
            break;  // Breaks the loop when the counter reaches 5
        }
    }

    return 0;
}
```



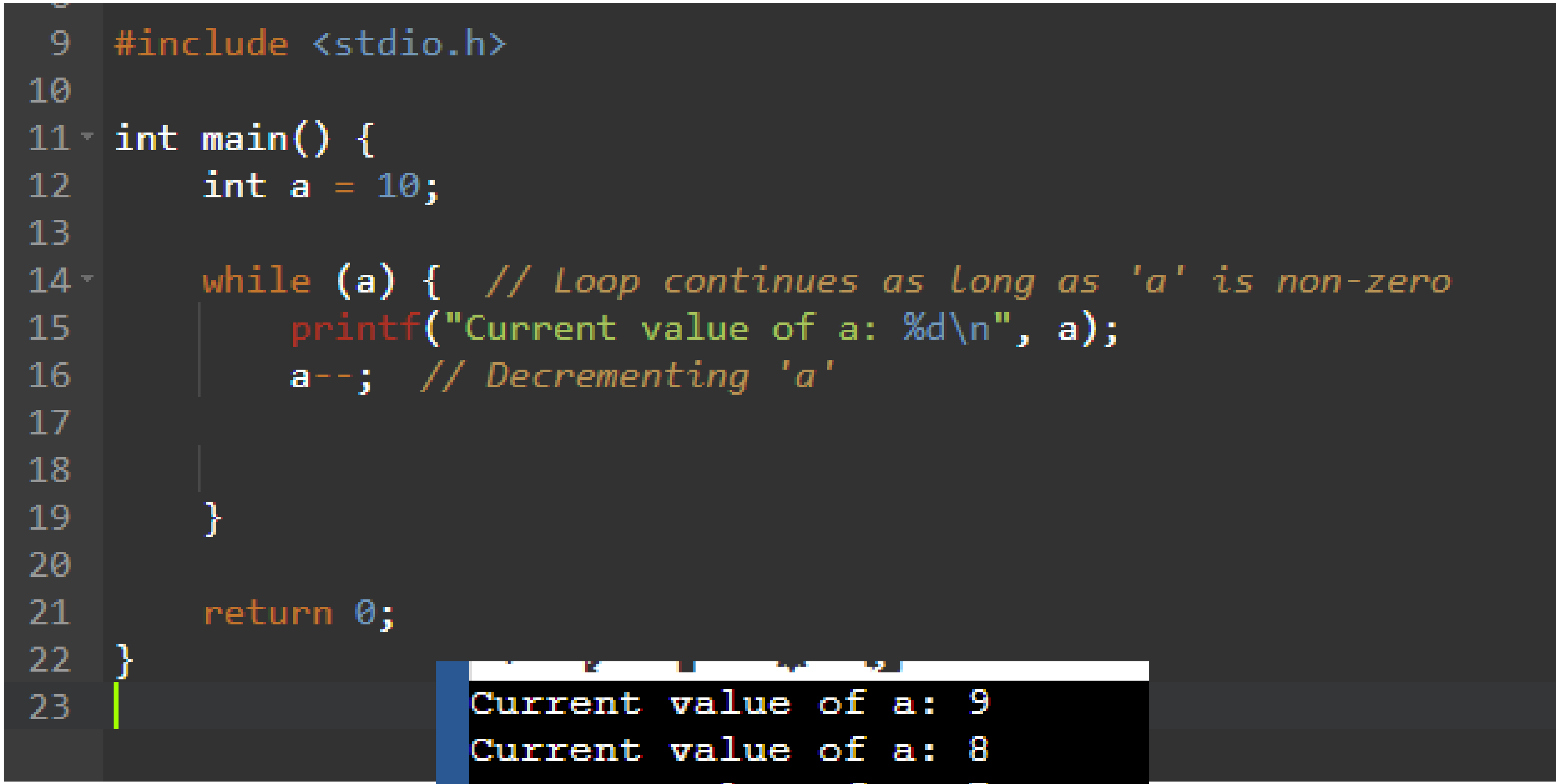
```
Iteration: 0
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
Exiting loop after 5 iterations.
```



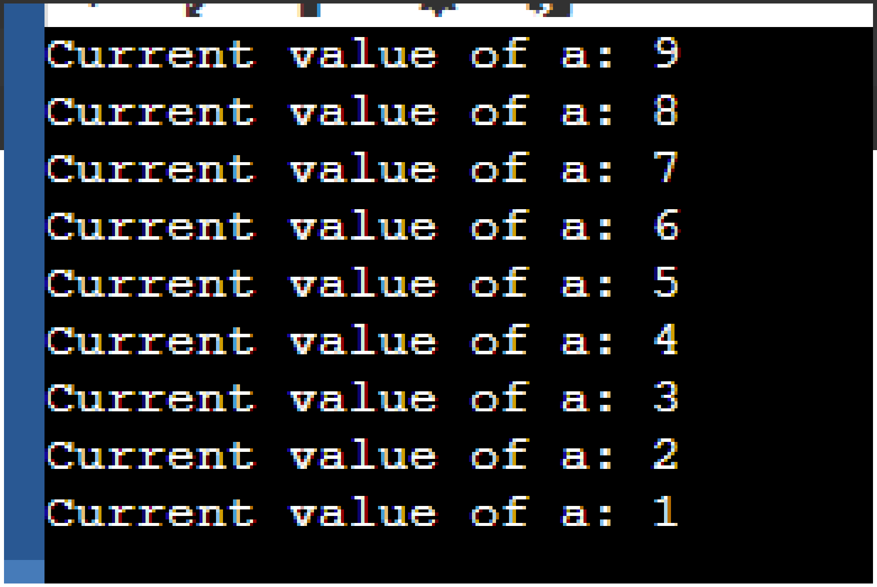
```c
int main() {
    int a = 10;

    while (a) {  // Loop continues as long as 'a' is non-zero
        printf("Current value of a: %d\n", a);
        a--;  // Decrementing 'a'

        if (a == 3) {
            printf("Stopping early as a reached 3.\n");
            break;  // Optionally breaking the loop early
        }
    }

    return 0;
}
```



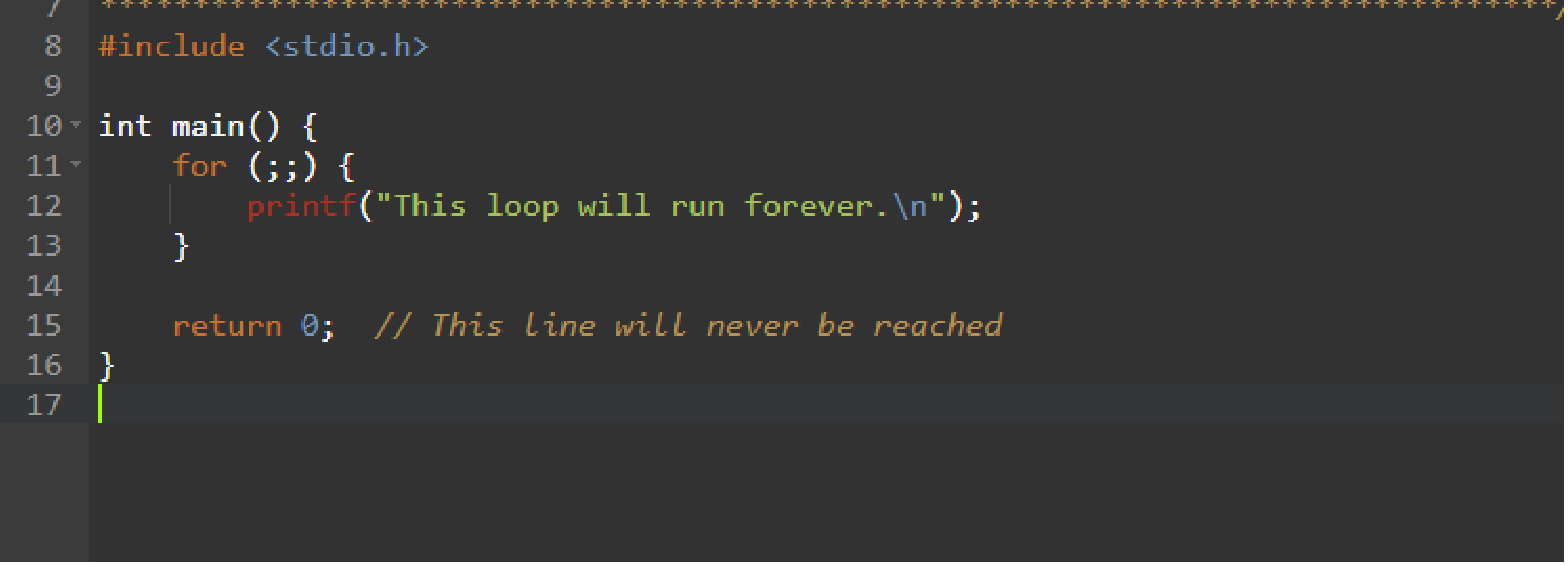
```
Iteration: 0
Iteration: 1
Iteration: 2
Iteration: 3
Iteration: 4
Exiting loop after 5 iterations.
```



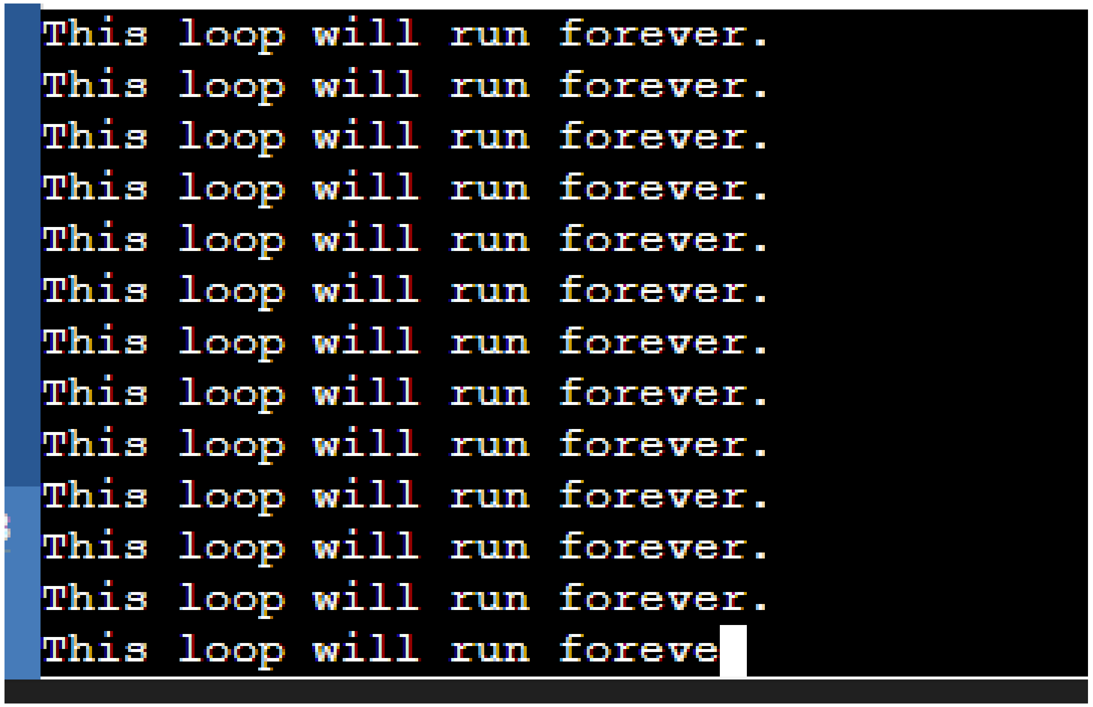
```c
#include <stdio.h>

int main() {
    int a = 10;

    while (a) {  // Loop continues as long as 'a' is non-zero
        printf("Current value of a: %d\n", a);
        a--;  // Decrementing 'a'


    }

    return 0;
}
```



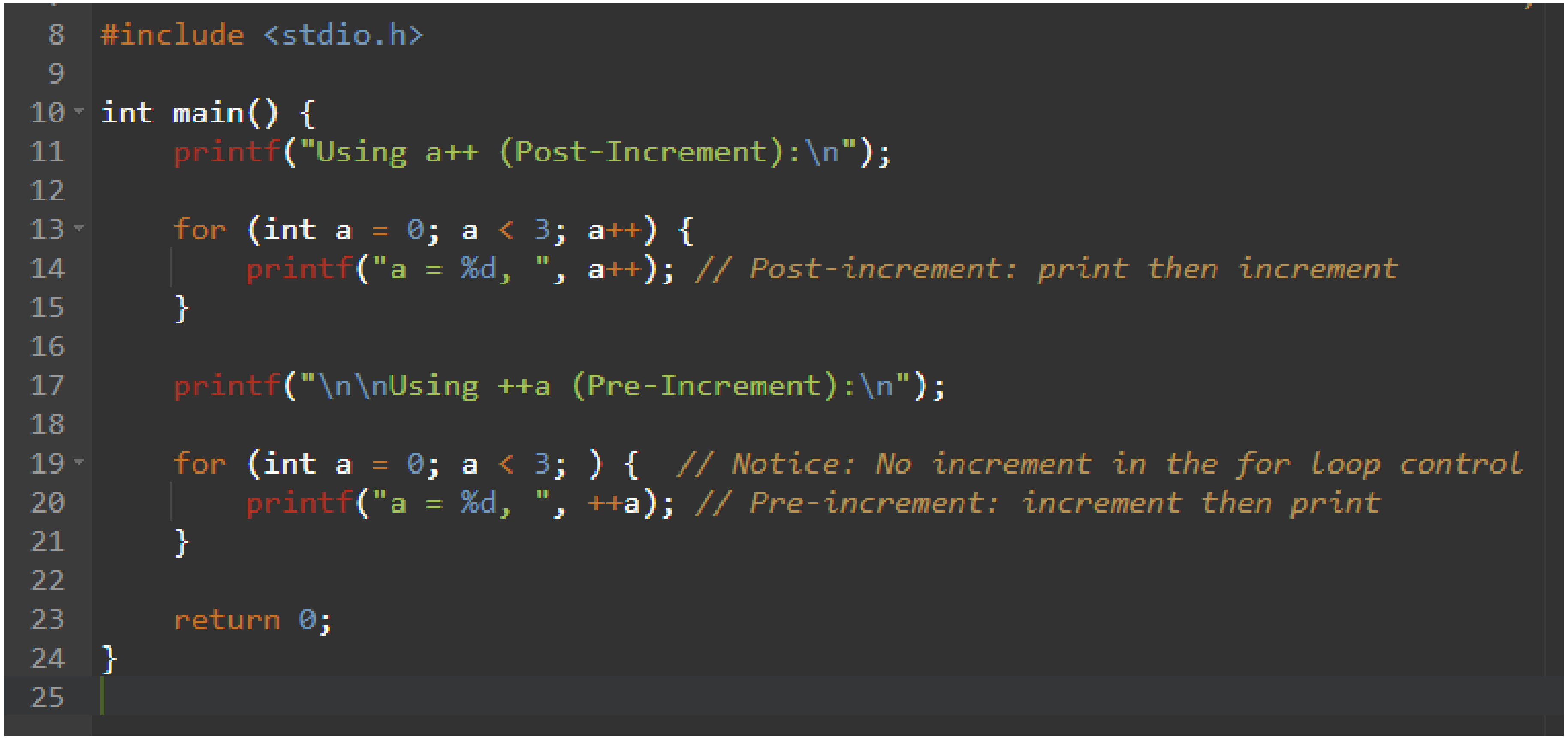
```
Current value of a: 9
Current value of a: 8
Current value of a: 7
Current value of a: 6
Current value of a: 5
Current value of a: 4
Current value of a: 3
Current value of a: 2
Current value of a: 1
```



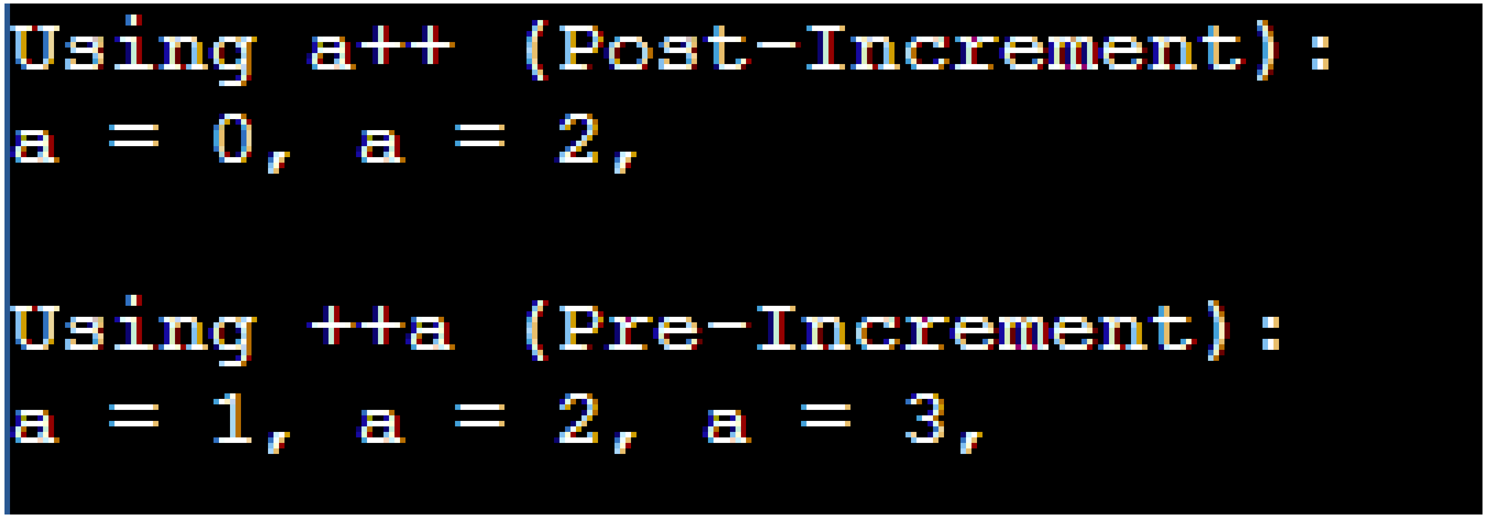
```c
#include <stdio.h>

int main() {
    for (;;) {
        printf("This loop will run forever.\n");
    }

    return 0;  // This line will never be reached
}
```



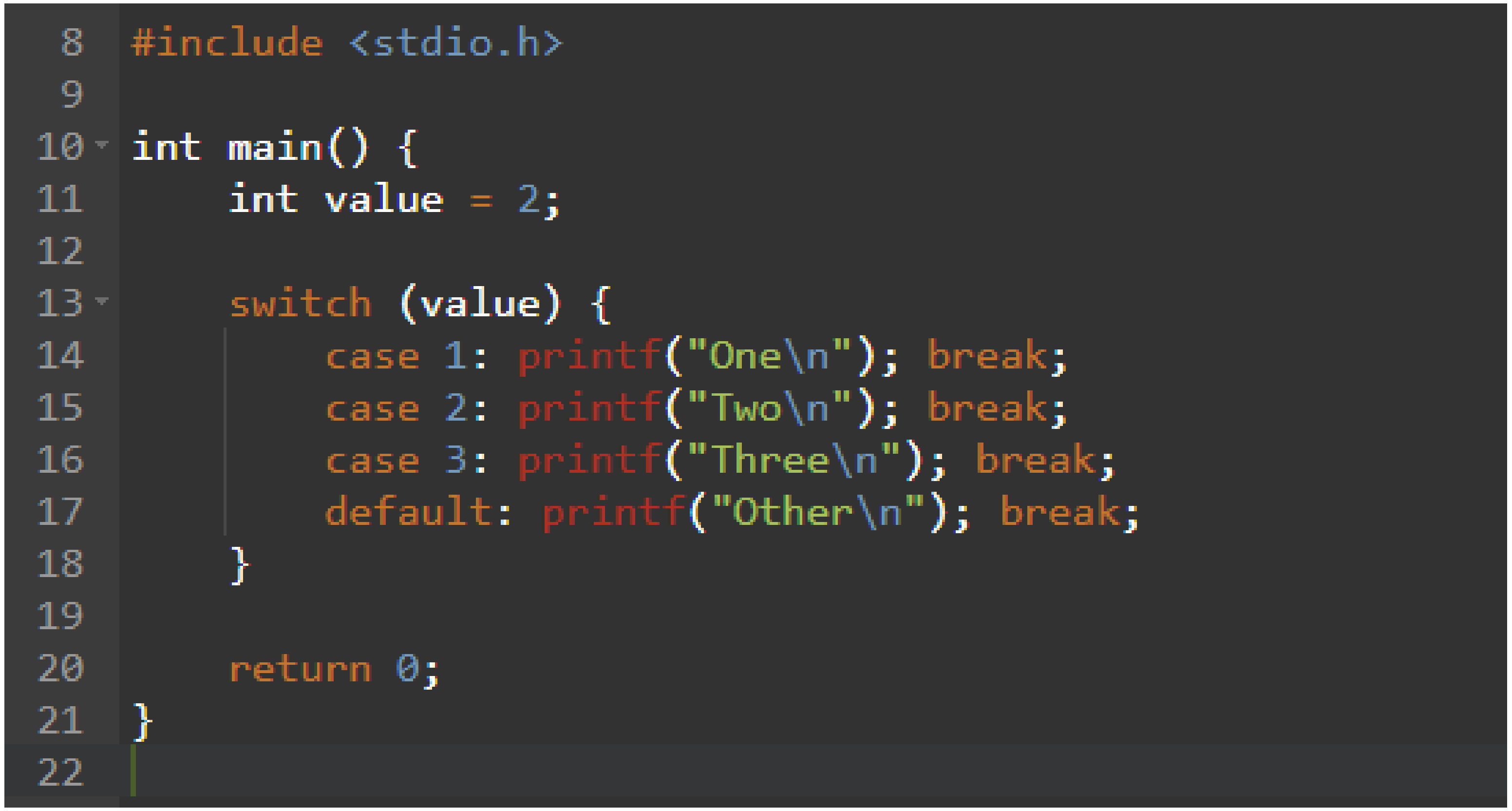
```
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run forever.
This loop will run foreve
```



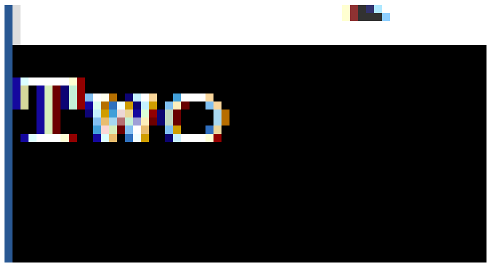
```c
#include <stdio.h>

int main() {
    printf("Using a++ (Post-Increment):\n");

    for (int a = 0; a < 3; a++) {
        printf("a = %d, ", a++); // Post-increment: print then increment
    }

    printf("\n\nUsing ++a (Pre-Increment):\n");

    for (int a = 0; a < 3; ) {  // Notice: No increment in the for loop control
        printf("a = %d, ", ++a); // Pre-increment: increment then print
    }

    return 0;
}
```



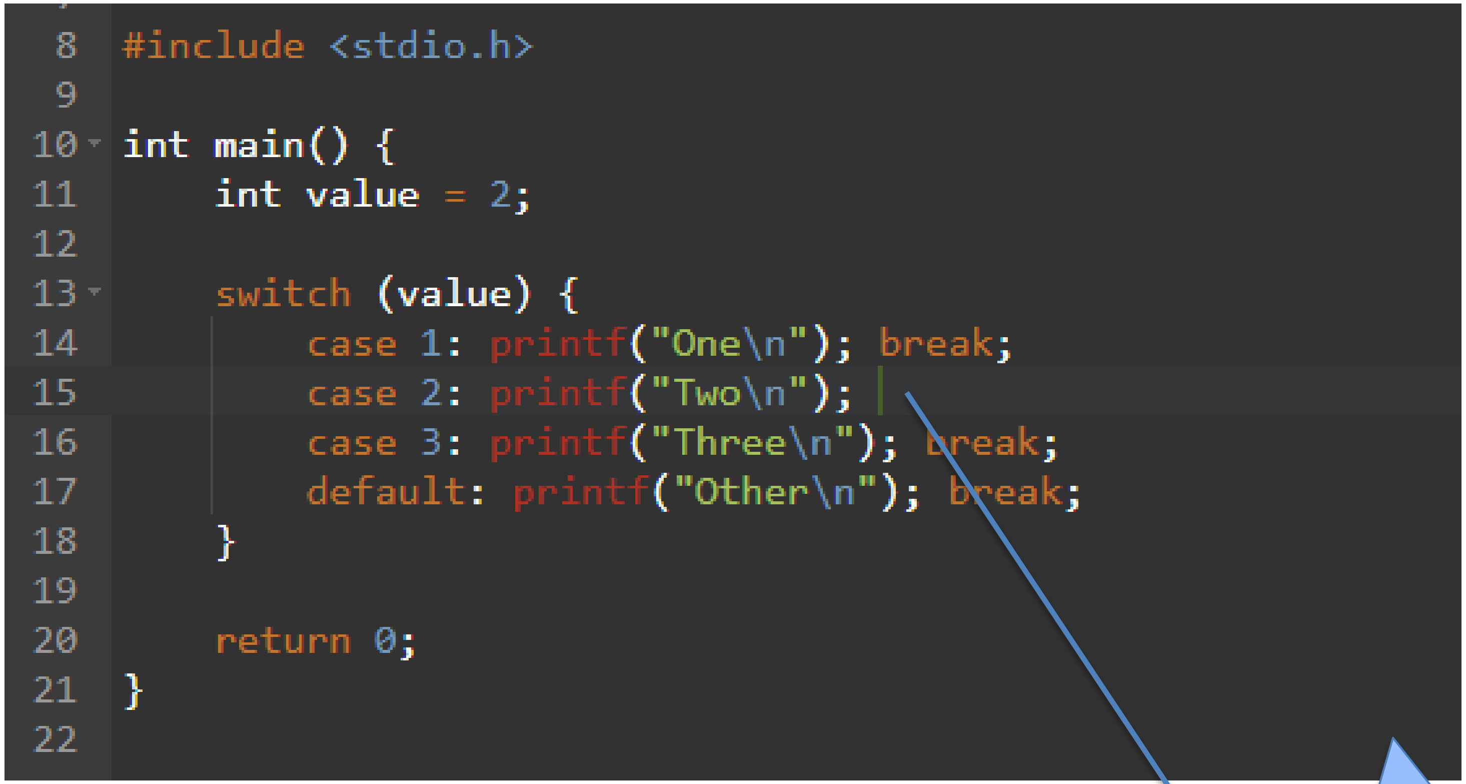
```
Using a++ (Post-Increment):
a = 0, a = 2,

Using ++a (Pre-Increment):
a = 1, a = 2, a = 3,
```



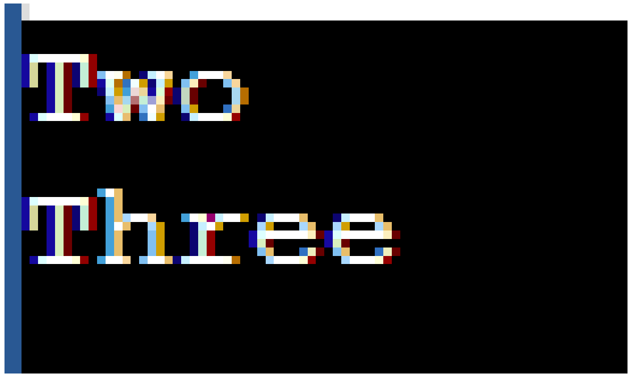
```c
#include <stdio.h>

int main() {
    int value = 2;

    switch (value) {
        case 1: printf("One\n"); break;
        case 2: printf("Two\n"); break;
        case 3: printf("Three\n"); break;
        default: printf("Other\n"); break;
    }

    return 0;
}
```



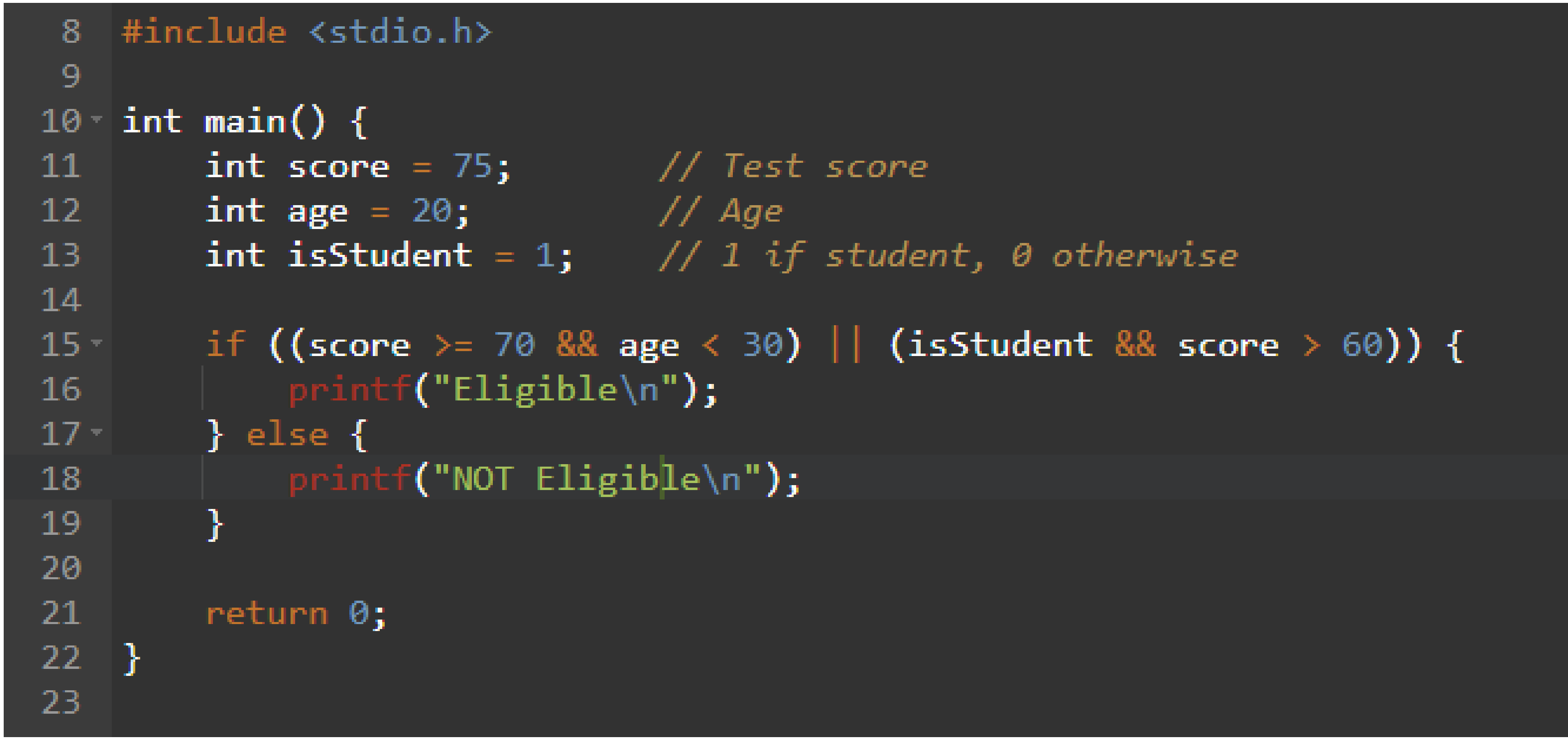
```
Two
```



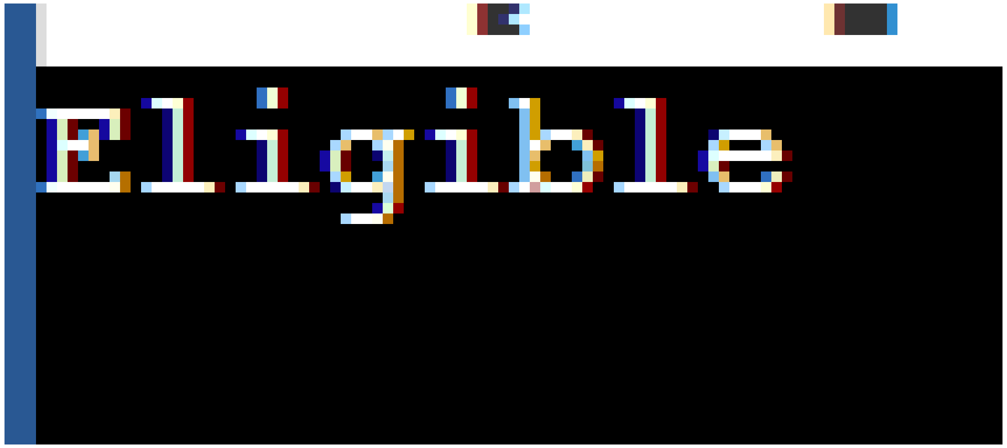
```c
#include <stdio.h>

int main() {
    int value = 2;

    switch (value) {
        case 1: printf("One\n"); break;
        case 2: printf("Two\n");
        case 3: printf("Three\n"); break;
        default: printf("Other\n"); break;
    }

    return 0;
}
```



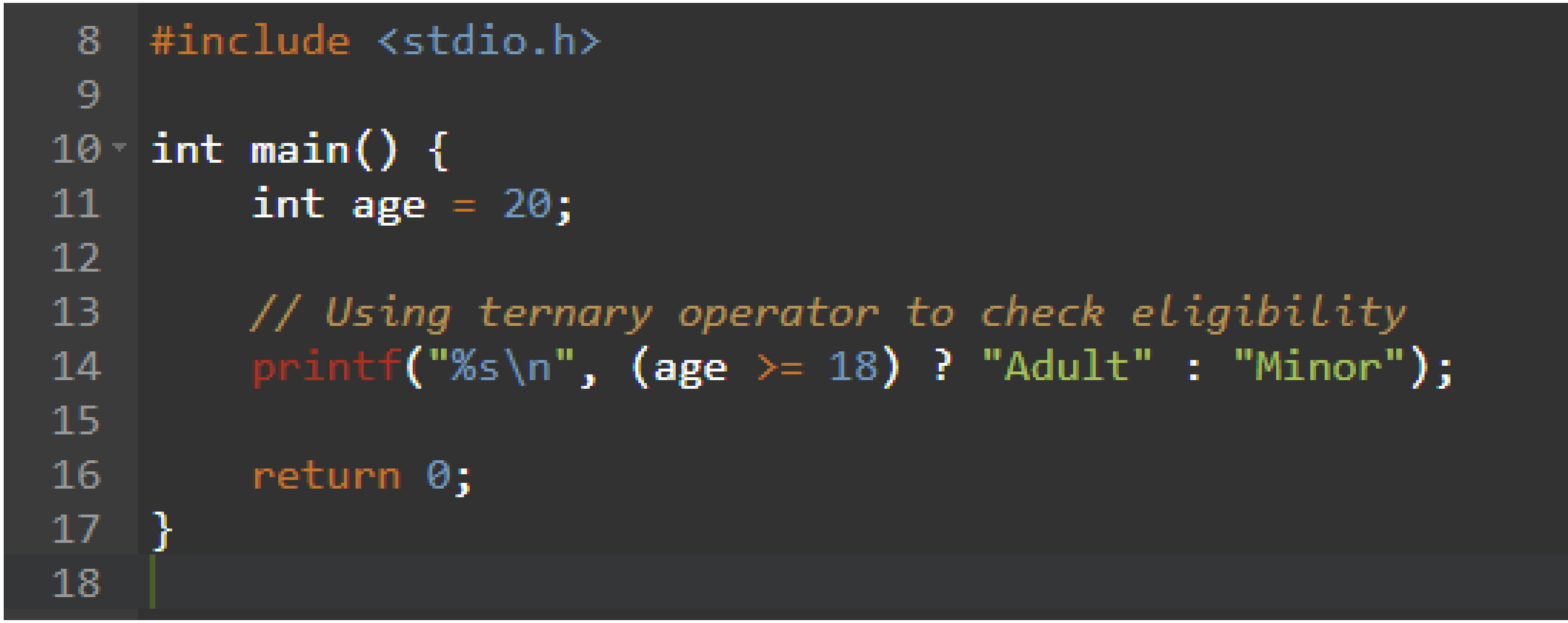
```
Two
Three
```



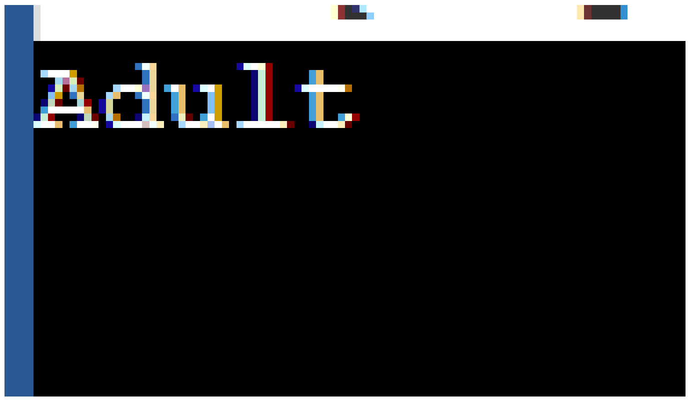
```c
#include <stdio.h>

int main() {
    int score = 75;         // Test score
    int age = 20;           // Age
    int isStudent = 1;      // 1 if student, 0 otherwise

    if ((score >= 70 && age < 30) || (isStudent && score > 60)) {
        printf("Eligible\n");
    } else {
        printf("NOT Eligible\n");
    }

    return 0;
}
```



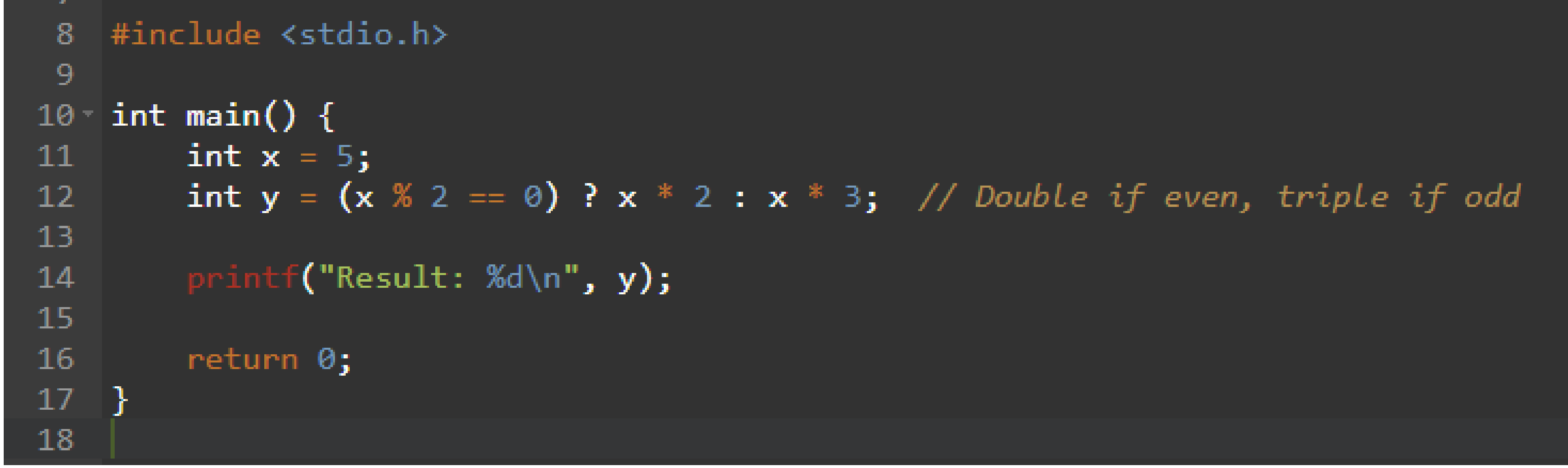
```
Eligible
```



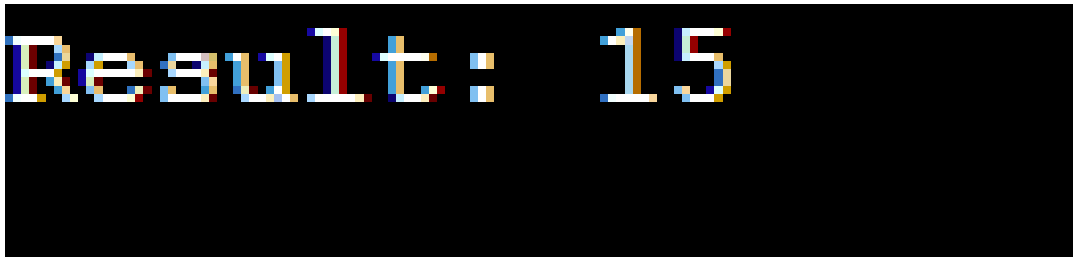
```c
#include <stdio.h>

int main() {
    int age = 20;

    // Using ternary operator to check eligibility
    printf("%s\n", (age >= 18) ? "Adult" : "Minor");

    return 0;
}

```



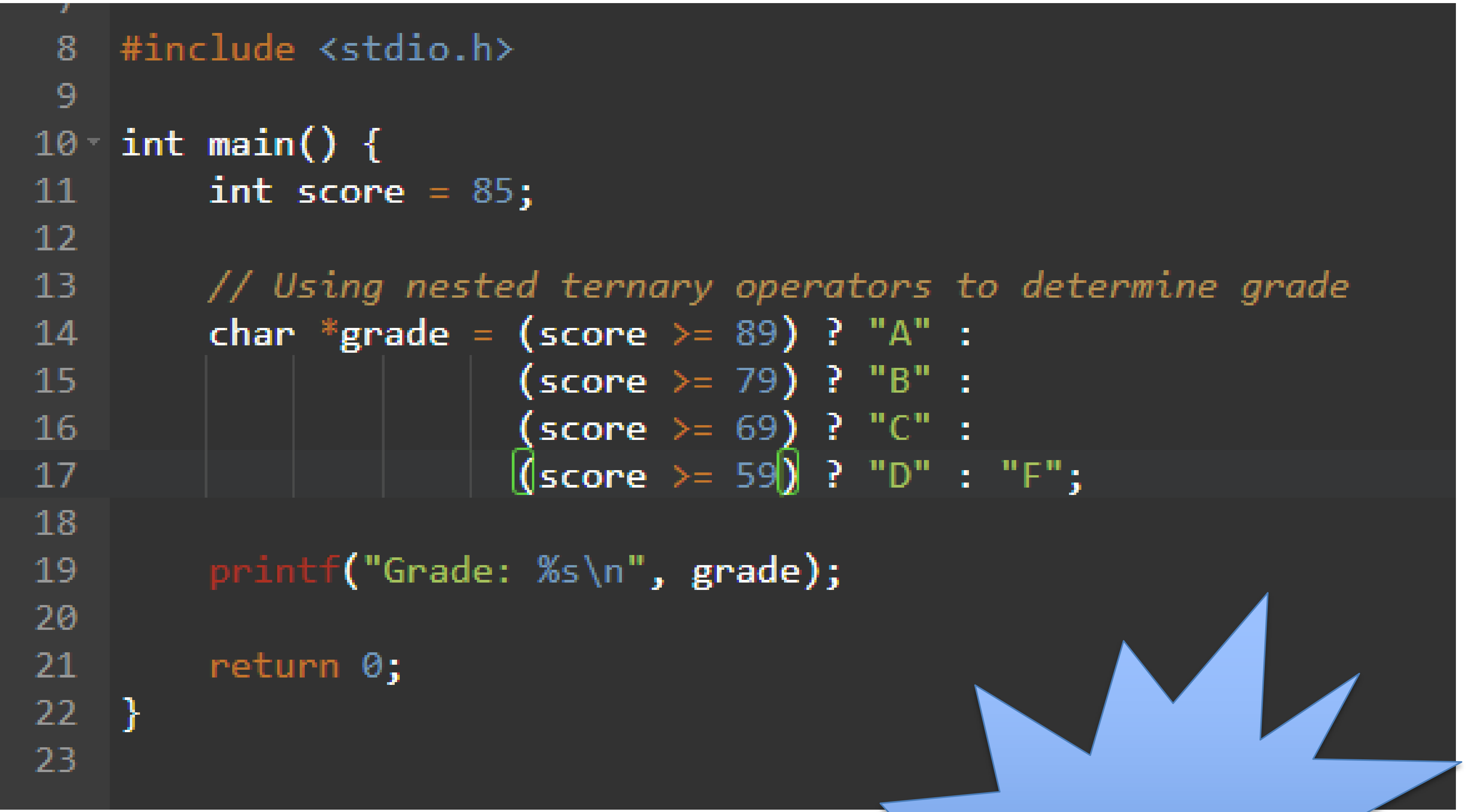
```
Adult
```



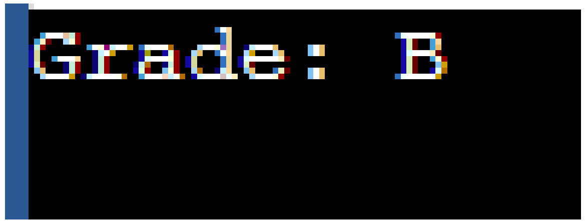
```c
#include <stdio.h>

int main() {
    int x = 5;
    int y = (x % 2 == 0) ? x * 2 : x * 3;  // Double if even, triple if odd

    printf("Result: %d\n", y);

    return 0;
}
```



```
Result: 15
```



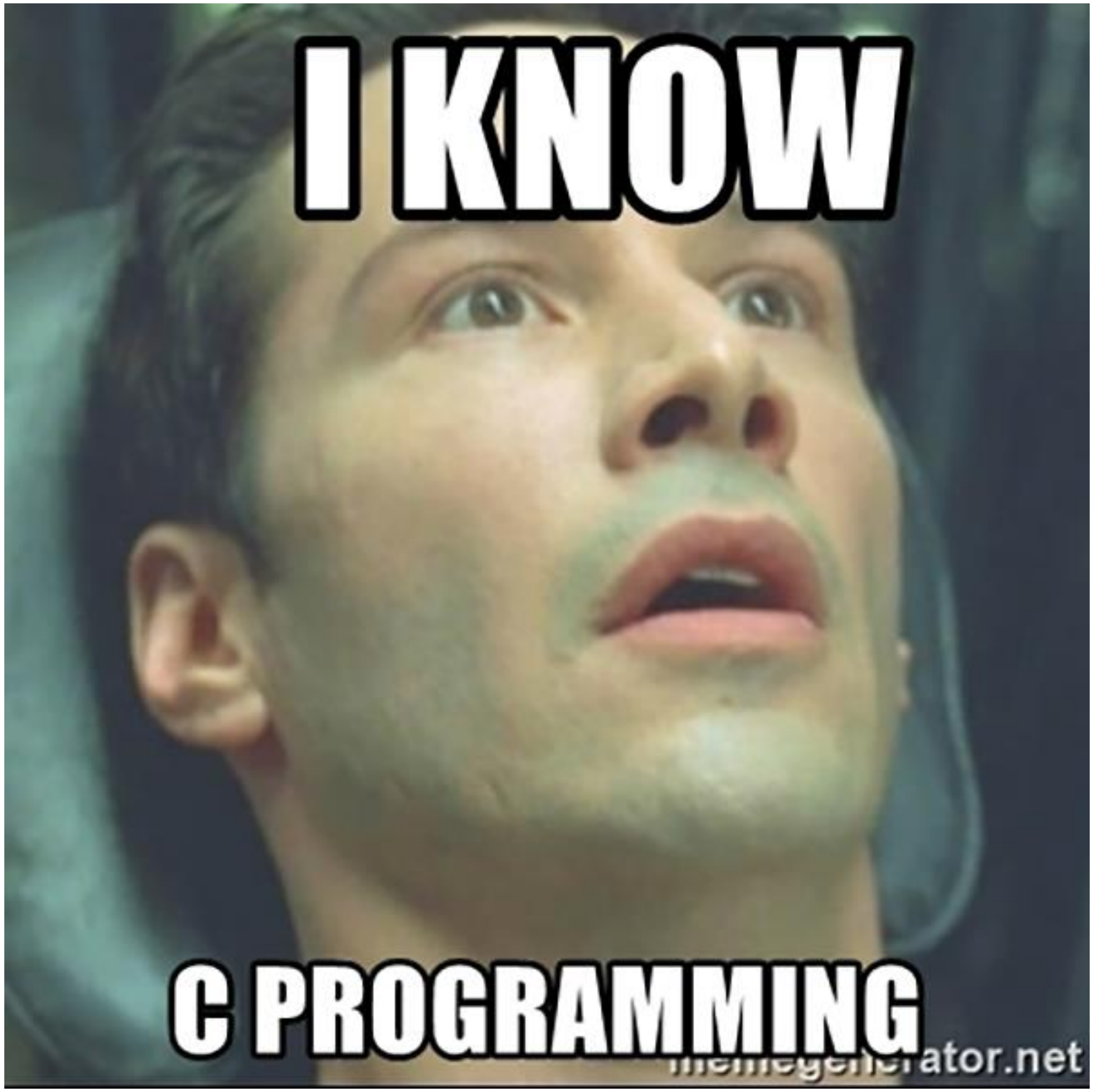
```c
#include <stdio.h>

int main() {
    int score = 85;

    // Using nested ternary operators to determine grade
    char *grade = (score >= 89) ? "A" :
                  (score >= 79) ? "B" :
                  (score >= 69) ? "C" :
                  (score >= 59) ? "D" : "F";

    printf("Grade: %s\n", grade);

    return 0;
}
```



```
Grade: B
```

Your final grades ☺

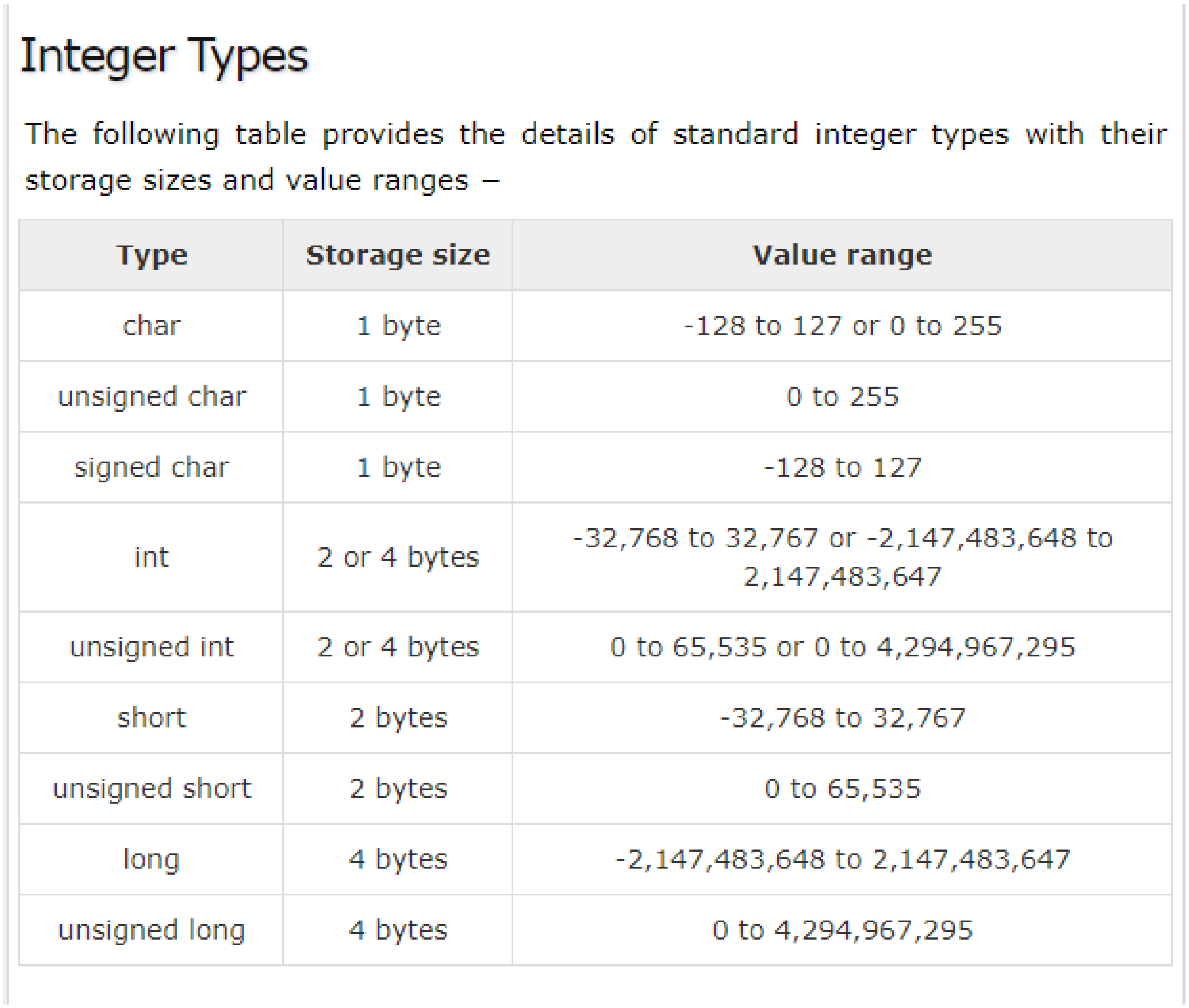
Not yet….



References

- https://www.tutorialspoint.com/cprogramming/c_constants.htm
- C From Theory to Practice - 2nd edition, Nikolaos D. Tselikas and George S. Tselikis

Data Types in C



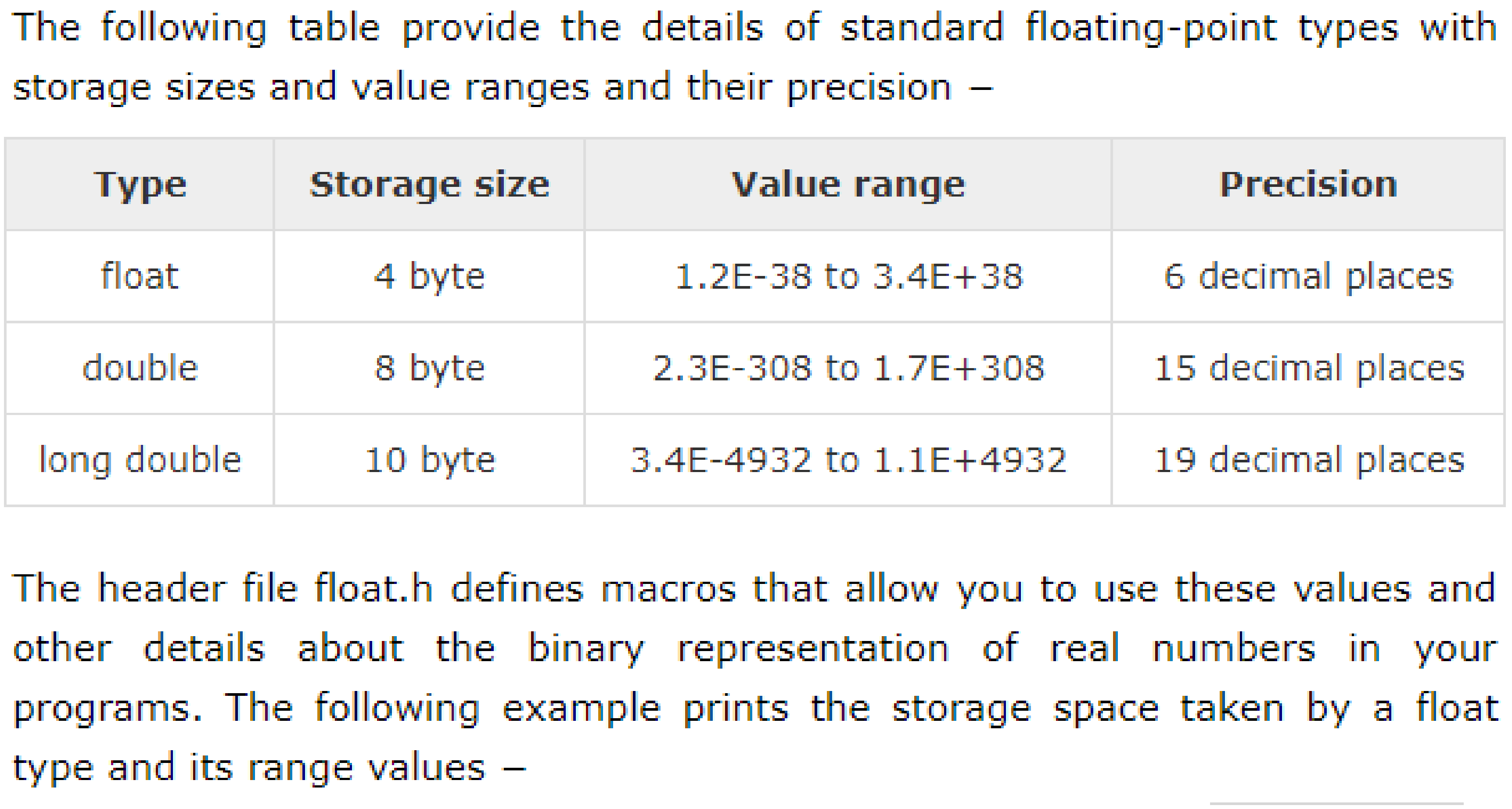
Integer Types

The following table provides the details of standard integer types with their storage sizes and value ranges –

Type	Storage size	Value range
char	1 byte	-128 to 127 or 0 to 255
unsigned char	1 byte	0 to 255
signed char	1 byte	-128 to 127
int	2 or 4 bytes	-32,768 to 32,767 or -2,147,483,648 to 2,147,483,647
unsigned int	2 or 4 bytes	0 to 65,535 or 0 to 4,294,967,295
short	2 bytes	-32,768 to 32,767
unsigned short	2 bytes	0 to 65,535
long	4 bytes	-2,147,483,648 to 2,147,483,647
unsigned long	4 bytes	0 to 4,294,967,295

https://www.tutorialspoint.com/cprogramming/c_data_types.htm

Floating-Point Types



The following table provide the details of standard floating-point types with storage sizes and value ranges and their precision –

Type	Storage size	Value range	Precision
float	4 byte	1.2E-38 to 3.4E+38	6 decimal places
double	8 byte	2.3E-308 to 1.7E+308	15 decimal places
long double	10 byte	3.4E-4932 to 1.1E+4932	19 decimal places

The header file float.h defines macros that allow you to use these values and other details about the binary representation of real numbers in your programs. The following example prints the storage space taken by a float type and its range values –

https://www.tutorialspoint.com/cprogramming/c_data_types.htm

C - Type Casting

- https://www.tutorialspoint.com/tpcg.php?p=LlUZrX

- https://www.tutorialspoint.com/tpcg.php?p=VkqrXY

- https://www.tutorialspoint.com/tpcg.php?p=MImFCX

https://https://www.tutorialspoint.com/cprogramming/c_type_casting.htm