CS101C: Introduction to Programming (Using C)
Autumn 2025

Nikhil Hegde
Achyut Mani Tripathi

Week3: Operator precedence, accepting input from terminal, if-else

So far..

- Printing on the terminal (printf)
- Data Types (int, float, double, char),
- Constants, Variables and their initialization using constants
- Operators and related background (bit representation)
  - Arithmetic (+, -, *, /, %)
  - Relational (==, !=, >, <, >=, <=)
  - Assignment (=, +=, -=, *=, /=, %, <<=, >>=, &=, ^=, |=)
  - Increment / Decrement (++, --)
  - Special: ternary, sizeof, comma
  - Logical (&&, ||, !)
  - Bitwise (&, |, ^, ~, <<, >>)

Today's class (18/8/2025)

- Operator precedence
- Accepting input from terminal
- C program to demonstrate:
  - Operator precedence
  - Accept input from terminal using scanf

Precedence	Operator	Description	Associativity
1	++ --	Suffix/postfix increment and decrement	Left-to-right
	()	Function call	
	[]	Array subscripting	
	.	Structure and union member access	
	->	Structure and union member access through pointer	
	(type){list}	Compound literal(C99)	
2	++ --	Prefix increment and decrement[note 1]	Right-to-left
	+ -	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(type)	Cast	
	*	Indirection (dereference)	
	&	Address-of	
	sizeof	Size-of[note 2]	
	_Alignof	Alignment requirement(C11)	
3	* / %	Multiplication, division, and remainder	Left-to-right
4	+ -	Addition and subtraction	
5	<< >>	Bitwise left shift and right shift	
6	< <=	For relational operators < and ≤ respectively	
	> >=	For relational operators > and ≥ respectively	
7	== !=	For relational = and ≠ respectively	
8	&	Bitwise AND	
9	^	Bitwise XOR (exclusive or)	
10	|	Bitwise OR (inclusive or)	
11	&&	Logical AND	
12	||	Logical OR	
13	?:	Ternary conditional[note 3]	Right-to-left
14[note 4]	=	Simple assignment	
	+= -=	Assignment by sum and difference	
	*= /= %=	Assignment by product, quotient, and remainder	
	<<= >>=	Assignment by bitwise left shift and right shift	
	&= ^= |=	Assignment by bitwise AND, XOR, and OR	
15	,	Comma	Left-to-right

Operator Precedence and Associativity

a=10, b=20, c=5. Evaluate:

d=a+b*c

d=a*b/c

d=b<a*c

d+=a=b

d+=a?b:c;

d=a+-b

challenge Q: d=++a+-b;?

4

```c
int main(){
    int a=10, b=20, c=5, d;
    d=a+b*c;
    printf("result of d=a+b*c: %d\n", d);
    d=a*b/c;
    printf("result of d=a*b/c: %d\n", d);
    d=b<a*c;
    printf("result of d=b<a*c: %d\n", d);
    d=0;
    d+=a?b:c;
    printf("result of d+=a?b:c %d\n", d);
    d=0;
    d+=a=b;
    printf("result of d+=a=b: %d. current value of a: %d\n", d, a);
    a=10;
    d=a+-b;
    printf("result of d=a+-b: %d.\n", d);
}
```

Sample Program - Operator Precedence

Scanf - a way to accept user input

```
int main() {
    int a;
    scanf("%d",&a);
    printf("%d",a);
}
```

- Reads user input from terminal (`stdin`) and stores in variables
- **&** - address of operator.
- **Arguments of scanf**: 1st = string, 2nd and subsequent (if present) = address of variables. How do you know if second and subsequent arguments are present?

Today's class (20/8/2025)

- Control-flow: `if-else`
- C program to demonstrate:
  - How the flow of execution can be controlled using the `if-else` construct

Control flow with `if-else`

- So far you have seen flow of control, where an instruction on a line executed after the instruction on the previous line.
- *The `if-else` construct in C changes that* i.e. the previously executed instruction need not necessarily be the one on the previous line.
- `E.g.` Execution of lines
- 4 not preceded by 3
- 6 not necessarily after 4

```
0: scanf("%d",&age);

1: if(age<18){

2: printf("cannot drive");

3: } else {

4: printf("drive safe");

5:}

6: printf("bye");
```

Control flow with `if-else`

- Curly braces {} can be omitted if you have a single statement within if-block / else-block
- The `else` part can be skipped. But if you have `else` part, you must have `if` part
- **Meaning of** `if-else`:

  If the condition is true, execute the if-block

  Otherwise execute the else-block

Which operator allowed conditional execution of an expression?

```
if(condition){

    statement1;

    ..

    statementn;

} else {

    statement1;

    ..

    statementn;

}
```

Control flow with `if-else`

- Can have `if-else` within another `if` and/or `else`
- New lines can be skipped. But having new lines makes your code readable.
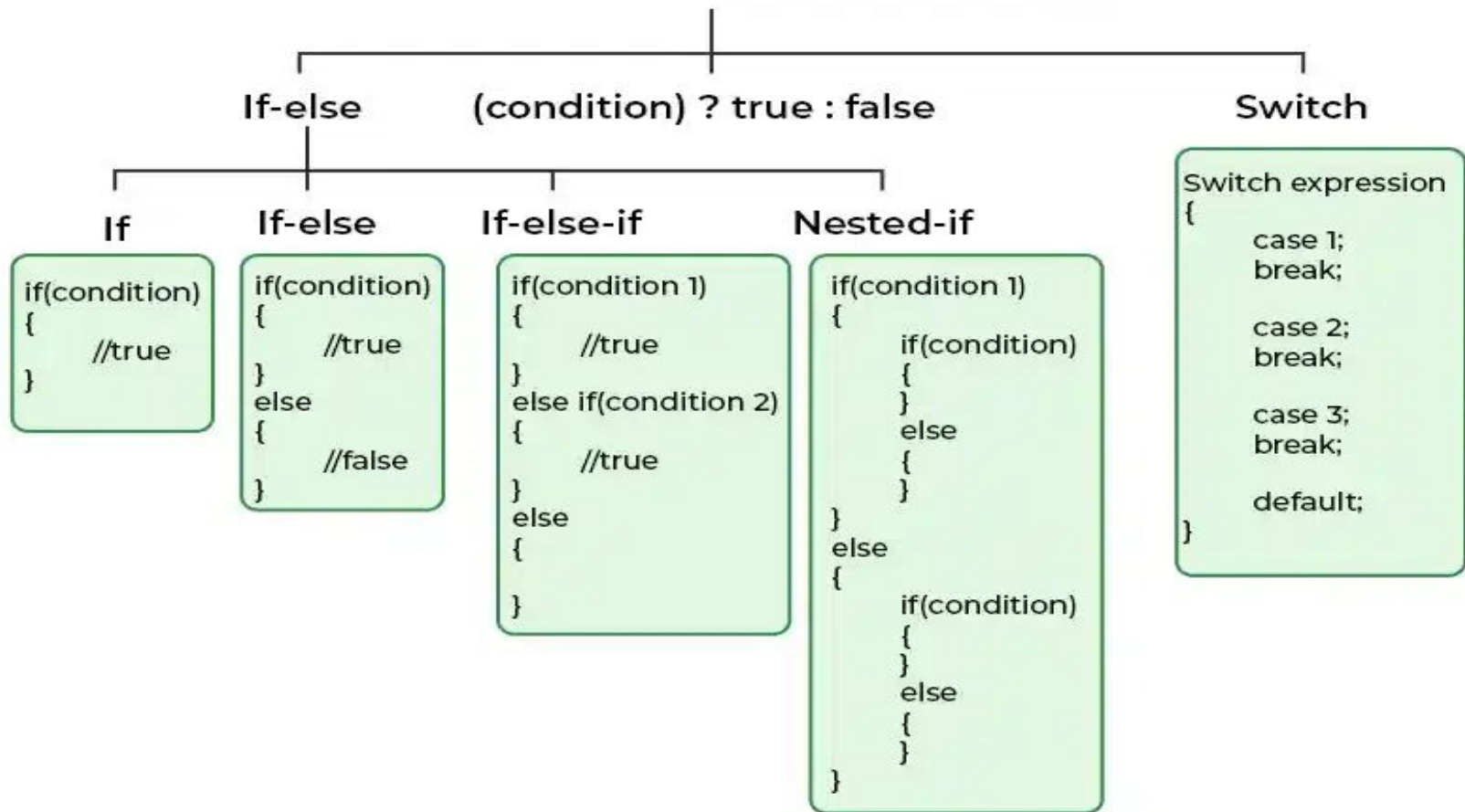
```
if(a<10){

    if(a>5){x=10;}

    else{x=100;}

} else {

    if(a>5){x=20;}

    else{x=200;}

}
```

Control flow with `if-else`

- There is `else-if` as well:
- The last else part is optional

```
if(a < 10){

    x=10;

} else if (a < 20){

    x=100;

} else if (a<30){

    x=1000;

} else {

    x=10000;

}
```

Conditional Statements in C

If-else   (condition) ? true : false   ## Switch

If
```
if(condition)
{
     //true
}
```

If-else
```
if(condition)
{
     //true
}
else
{
     //false
}
```

If-else-if
```
if(condition 1)
{
     //true
}
else if(condition 2)
{
     //true
}
else
{

}
```

Nested-if
```
if(condition 1)
{
     if(condition)
     {
     }
     else
     {
     }
}
else
{
     if(condition)
     {
     }
     else
     {
     }
}
```

Switch expression
```
Switch expression
{
     case 1;
     break;

     case 2;
     break;

     case 3;
     break;

     default;
}
```

Control flow with `switch-case`

- `if-else` gives you two-way branching
- with `else-if` you get multi-way branching
- `switch-case` construct in C also gives you multi-way branching.

```
switch(class){

    case 1: printf("MON");

            break;

    case 2: printf("WED");

            break;

    case 3: printf("FRI");

            break;

    default: printf("no
class");

}
```