Programming Foundations

C Programming - Syntax Overview

Rahul Ekbote, Sep 2022

Keywords

What?

- C Keywords are reserved words that can’t be redefined or used for any other meaning.

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

https://en.cppreference.com/w/c/keyword

Comments

Add Code Documentation

- Purpose of comments in C (and any Programming)
    - Add easy-to-understand text for other programmers

Location	Purpose
Beginning of the Source file	Explain the purpose of the .c or the .h file
Beginning of the Function	• Purpose • Input Parameters • Output Value
Inside the Function Code	• Single or multi-line comments • Explain the purpose of the code construct
Inside the Function Code - at the end of the source code line	Add a small single line comment at the end of the C code line to explain something in brief

code\syntax\comments.c

Simple Data Types - 1

char, short, int, long, double, float (note - no boolean data type)

- Integral types and their typical sizes (sizes may vary based on the architecture). Each can be signed and unsigned
  - char - 1 Byte (8 bits) - Best Example - ASCII table
  - short - 2 Bytes (16 bits)
  - int - 4 bytes (32 bits)
  - long - 4 or 8 bytes (32 or 64 bits)

src\syntax\integralTypes.c

Simple Data Types - 2

Floating Point/Real Numbers

- Floating Point numbers are represented by IEEE 754.
- Typical sizes (can vary based on the underling architecture)
    - float - 4 bytes
    - double - 8 bytes
- only signed floating points. NO UNSIGNED FLOATING POINT in C

src\syntax\floatingPointTypes.c

Boolean Data Type and Void

Supported in C99 standard only

- Boolean data types contain either TRUE or FALSE (1 or 0)
- A very useful type but not supported in C until C99 Language standard
- To use it, a standard header file stdbool.h must be included

src\syntax\booleanType.c

- Void (keyword: void) is a standard data type which indicates no special value. Used to indicate that no data is being referred

C Variables - 1

Data storage for the C Programs

- C Variables store data
- Variable - literally means a changeable(varying) entity
- Variables are names assigned to the memory location
- Variables have various data types
  - Standard Data Types - covered earlier
  - User Defined Data Types - structs/unions - to be covered later

C Variables - 2 continued

Variable naming rules -

- Must be a meaningful name in a given programming context
- Must not begin with a number/digit
- OK to include digits, alphabets and underscore(_)
- Can’t be C reserved words/Keywords
- No spaces/blanks in the name

src\syntax\vars.c

Variable Scopes - 1

Global Scope

- Variables can be accessed only within their scopes
- Global Scope -
    - Variables defined in the Global Scope are accessible in all the functions within that file (and also other files)
    - Should be avoided OR used extremely rarely
    - Cause of various types of bugs in the program.

src/syntax/globalScope.c

Variable Scopes - 2

Local Scope

- Variables defined within the function block have local scope
- This scope is ephemeral (temporary) and all the variables go out-of-scope when the function returns
- Variables with the same name defined in two different functions have only local scope and do not affect each other in any way.
- Most C code would use the Local Scope (over Global Scope)

src/syntax/localScope.c

Operators
Arithmetic

- Simple Operators

+	Addition	a + b
-	Subtraction	a - b
*	Multiply	a * b
/	Divide	a / b
%	Modulo (Remainder)	a % b
++	Prefix and Postfix Increment	a++, ++a
- -	Prefix and Postfix Decrement	a—, —a

No Source Code, Try your own

Operators

Relational

- Comparison Operators

==	Equals	a == b
!=	Not Equals	a != b
<	Less Than	a < b
>	Greater Than	a > b
<=	Less Than OR Equal	a <= b
>=	Greater Than Or Equal	a >= b

src\syntax\compareOps.c

Operators

Logical

- Use in conditions that involve checking AND-OR logic
- For example:
  - checking an age between a certain range
  - checking if a student has has scored enough to move to the next semester
- Multiple AND-OR conditions can be combined to handle complex logic
- Used typically with an if-else conditional blocks

AND	&&	• a && b • a && b && c
OR	||	• a || b • a || b && c
NOT	!	• !a • !(a && b)

src/syntax/logicalOps.c

Operators

Bitwise

- AND-OR-XOR operations are done bit-wise.
- AND: &
- OR: |
- XOR: ^
- Extremely Useful in Low-Level Programming

A	**B**	A & B	A | B	A ^ B
0	**0**	0	0	0
0	**1**	0	1	1
1	**0**	0	1	1
1	**1**	1	1	0

No Source code: Advanced Topic: Covering later

Assignment

Statements that assign RHS to LHS

- Assignments and Shortcuts

=	a = b	Set “a” to the value of “b”
+=	a += b	Short for: a = (a + b)
-=	a -= b	Short for: a = (a - b)
*=	a *= b	Short for: a = (a * b)
/=	a /= b	Short for: a = (a / b)
%=	a %= b	Short for: a = (a % b)
<<=	a <<= b	a = a << b (Shift Left)
>>=	a >>= b	a = a >> b (Shift Right)
&=	a &= b	a = a & b (Bitwise AND)
^=	a ^= b	a = a ^ b (Bitwise XOR)
|=	a |= b	a = a | b (Bitwise OR)

src/syntax/assignmentOps.c

Important Other Operators

Very Important

- Highly Important Ops

sizeof()	a = sizeof(b)	a is set to the number of bytes that b would take in memory
? :	x = (a == b) ? c : d	• Evaluate a == b • If TRUE, set x to c • Else, set x to d
&	a = &b;	• & - Reference Operator • Sets b to the memory address of b
*	x = *p	• p is treated as a memory address • x is set to the contents of p

src/syntax/impOps.c

Operator Precedence

Look at online reference for detailed table of operator precedence

- Operators have precedence
- For example - express 2 + 3 * 4 = ?
  - 20 OR 14
- Operator Precedence is important to make sure that the expressions in the C program can be interpreted and evaluated in correct order
- Parentheses used as a primary correction mechanism
- For example - (2+3) * 4 in the example above.

input/output/error

reading from the keyboard, writing to the screen

- OSs(Windows/Linux/Unix) maintain 3 standard sources/streams
- Keyboard - standard input - “stdin” in a C Program
- Whatever a C Program reads, it will be read from the keyboard
- Screen - standard output - “stdout” in a C Program
- Whatever a C program writes, appears on the screen
- The 3rd standard source is standard-error, “stderr” in a C Program
- Typically, stderr is also directed to the screen

printf() - 1

send output to the screen (stdout device)

- printf() is a C standard library function
- Its used to print something to the screen from your C Program
- To use it without getting any compiler warnings, you must include a header file at the top of your C source file using #include <stdio.h>
- The printf() function can take one or more parameters per the need. This type of function is also called as a “varargs” function because the number of parameters vary.
- Used with the format string as the 1st parameter

printf() - 2

Output

- Common format specifiers and Most Used Escape Characters
-

%d	Integer
%c	Single Character
%hu	Unsigned short
%hi	Signed short
%ld	Long in Decimal Format
%x	Hexadecimal

%o	Octal
%p	Pointer(Address/ Reference)
%f	Float
%e or %E	Floating Point Scientific Notation
%%	To print the % symbol itself

\n	Newline
\t	Tab
\’	Single Quote
\”	Double Quote
\\	Backslash

src/syntax/simplePrintf.c src/syntax/escapeChars.c

Simple Reading Input into a C Program

scanf()

- scanf() function allows the C Program to read a user input from the keyboard
- Because it's reading from the keyboard, scanf() needs a format specifier like in printf() and *also* a pointer to the location where to store the data that's read from the keyboard
- Format specifiers similar to the printf() function
- Little cumbersome to use because of preceding scanf()s and newlines

src/sytax/simpleScanf.c

Conditional Logic

if..else if..else

- Every meaningful program has conditional logic
- For example -
  - Student matches certain grades
  - Person is eligible for certain tax rate based on his/her age
  - If a date is in certain range, etc.

src/syntax/conditional-if.c
src/syntax/conditional-if-else.c
src/syntax/conditional-if-elseif.c

Conditional Logic
switch statement

- switch statement reduces complicated if..elseif code blocks
- switch statement makes the conditional logic readable
- switch statement uses ‘case’ keyword to define a valid condition
- code block that defines the case typically ends with a “break” keyword
- “default” keywords defines a default block of code that’s executed when no match occurs.

src/syntax/switch-1.c
src/syntax/switch-2-nobreak.c
src/syntax/switch-3-default.c

Iterations

for loop

- for loop offers an important way to implement iterations in C
- Basic syntax of the for loop
    - for(initializers; conditional; posterior) { }
- initializer -> Statements that are executed once before the for loop begins
- conditional -> Condition that’s executed once to decide if the code block should execute
- posterior -> Statements that execute after the code block executed.

Iterations

for loop examples

- simpleFor.c -> Simple implementation of a for loop.
- forLoopNoInitializer.c -> No Initializers
- forLoopNoPosterior.c -> No posterior statement block
- forLoopForever -> Code with neither the initializers nor the posterior
- forLoopMultiple.c -> Code with multiple inits and embedded for loops.

src/syntax/for_loop.c

Iterations

do..while

- do..while loops are similar to the while loops
- Key difference is when the boolean expression is evaluated
- while() {} -> condition evaluated before the code block execution
- do..while() {} -> code is executed at least once and the condition is evaluated at the end of that first iteration (and every other iteration)

C Preprocessor

#include

- All C preprocessor directive start with a hash symbol (#)
- They are the indication to the compiler that these are preprocessor directives and should be, therefore, processed before the C program is compiled
- The #include directive
    - asks the preprocessor to include a header file into the program
    - the header files provide prototype and type declarations (and not code)
    - The prototypes are used to define the function interface

src/syntax/hashInclude.c

C Preprocessor

#define

- #define directive defines manifest constants
- The #defined constant used in the C program is automatically *replaced* by its value
- This allows all the usages to be changed just by changing the manifest constant’s value in one place
- Therefore, this mechanism allows to reduce the clutter and possibility of bugs in the code

src/syntax/hashInclude.c