C Programming – Part 1

Aravind Prakash

aprakash@Binghamton.edu

Contents

- Introduction, History and Prelude
- Hello World
- Keywords and Types
- Data and Representation
- Operators and Expressions
- Decision Statement
- Loops

Contents

Introduction -- C

- Was developed in 1972 by Dennis Ritchie in Bell Labs
- Is an update of `B’ (basic combined programming) developed by Ken Thomson
- Is a compiled programming language
- Is a structured programming language
- Programs written in C can be fast and highly efficient
- Source files are *.c (like *.java and *.cpp) and header files are *.h (like interfaces in java and *.hpp in C++)

Compilation

- A compiler transforms a program source code into low-level representation (e.g., machine code).
  Example: gcc, clang, visual studio
- Compiler vs Interpreter

Compiler vs Interpreter (https://www.programiz.com/article/difference-compiler-interpreter)

Interpreter	Compiler
Translates program one statement at a time.	Scans the entire program and translates it as a whole into machine code.
It takes less amount of time to analyze the source code but the overall execution time is slower.	It takes large amount of time to analyze the source code but the overall execution time is comparatively faster.
No intermediate object code is generated, hence are memory efficient.	Generates intermediate object code which further requires linking, hence requires more memory.
Continues translating the program until the first error is met, in which case it stops. Hence debugging is easy.	It generates the error message only after scanning the whole program. Hence debugging is comparatively hard.
Programming language like Python, Ruby use interpreters.	Programming language like C, C++ use compilers.

Contents

Hello World

- `/* hello.c */`
- `#include <stdio.h> /* We want to use a declaration in stdio.h */`
- `/* main function is the entry point */`
- `int main () {`
  - `printf(“Hello World”);`
  - `return 0;`
  - `}`
- $ gcc hello.c
- $ ./a.out

printf and scanf Families

- `$ man printf; man scanf`
- printf function allows you to send output to standard out (monitor)
- scanf function reads input from standard input (keyboard)

Program to add two numbers

```
/* add.c */
#include <stdio.h>
int main() {
    int n1, n2, n3;  /* Identifiers */
    printf("Enter number 1: ");
    scanf("%d", &n1);
    printf("Enter number 2: ");
    scanf("%d", &n2);
    n3 = n1 + n2;
    printf("%d + %d = %d\n", n1, n2, n3);
}
```

Contents

- Introduction, History and Prelude
- Hello World
- Keywords and Types
- Data and Representation
- Operators and Expressions
- Decision Statement
- Loops

Identifiers

- Identifiers: are names you supply for variables, functions and other labels in your program.
- No two identifiers can be same.
- Identifiers are case sensitive.
- Identifier Rules:
  - First character in an identifier must be an alphabet (or underscore). That is, it can’t start with a number.
  - Must consist of only letters, digits or underscore.
  - Only first 31 characters are significant.
  - Can not use a keyword.
  - Must not contain white space.

Keywords

- Letters, Digits, Special Characters, White Space
- Keywords: are words that have special meaning to C compiler. They can not be used as identifiers.
- Keyword list: auto, double, int, struct, break, else, long, switch, case, enum, register, typedef, char, extern, return, union, const, float, short, unsigned, continue, for, signed, void, default, goto, sizeof, volatile, do, if, static, while.

Data Types (ANSI C)

- Constants: Numeric (integer, real), Character (single, multi)

Data Types (ANSI C)

Type	Size	Range
signed char	1	-128 to 127
Unsigned char	1	0 to 255
Short signed int	2	-32768 to 32767
Short unsigned int	2	0 to 65535
long signed int	4	-2147483648 to 2147483647
Long unsigned int	4	0 to 4294967295
Float	4	-3.4e38 to +3.4e38
Double	8	-1.7e308 to +1.7e308
Long double	10	-1.7e4932 to +1.7e4932

Contents

- Introduction, History and Prelude
- Hello World
- Keywords and Types
- Data and Representation
- Operators and Expressions
- Decision Statement
- Loops

Operators

- Operator indicates an operation to be performed on data that yields a value.
  - Arithmetic (+, -, *, /, %) -- Binary
  - Relational (>, < , ==, >=, <=, !=) – Binary
  - Logical (&&, ||, !)
  - Increment and decrement (++, --)
  - Assignment (=, +=, -=, *=, /=, %=, …)
  - Bitwise (&, |, ^, >>, <<)
  - Comma (,)
  - Conditional (?:)
  - Misc: sizeof, address of (&), pointer (*), dot (.) and points to (->)

Operator precedence and associativity

- K&R 2.12
  - Not required to remember, but need to get right. Use parenthesis to disambiguate when uncertain.

Escape characters

- \a alert, audible alarm, bell
- \b Backspace
- \f Form feed
- **\n New line, carriage return and line feed**
- \o octal constant
- \r carriage return, no line feed
- **\t horizontal tab**
- \v vertical tab
- \x hex constant
- \” quote character
- \’ apostrophe character
- \\ backslash character

Format specifiers for printf and scanf families

- **%c a single character**, char
- **%d a decimal number**, int
- %hd is for short %ld is for long
- %e a floating point number in scientific notation, %le is for double, %Le is for long double
- **%f a floating point number** with decimal point %10.4f 10 wide.dddd, %lf is for double, %Lf is for long double
- %g a floating point number, %f or %e as needed, %G for capital E %lg is for double, %Lg is for long double
- %i an integer, int %hi is for short int, %li is for long int
- %n pointer to integer to receive number of characters so far, int *i
- %o an unsigned octal number, unsigned int %ho and %lo for short and long
- **%p a pointer, void **x**
- **%s a string (must be null terminated !)**, use %s for scanf
- **%u an unsigned decimal integer** (printf only)
- **%x a hexadecimal number**, %X for capital ABCDEF.

Contents

- Introduction, History and Prelude
- Hello World
- Keywords and Types
- Data and Representation
- Operators and Expressions
- Decision Statement
- Loops

Conditional/Decision statements

- Used to alter control flow of a program
  - If
  - If-else
  - Switch
  - Ternary operator

if statement

- If(expr) statement;
- If(expr) { statement1; statement2; … }
  - where expr is a Boolean
- 0 is False, everything else is True

If-else statement

- If(expr)
  - statement1;
- else
  - Statement2;
- If(expr1) { statement1; statement2; … }
- Else if {statement3; statement4; …}
- Else {statement5; … }

Nested if

- If(expr1)
  - if(expr2) …
- /* nested.c */
- …
- int main () {
  - Int a, b, c;
  - …
  - If(a>b) {
    - if(a>c) {printf("A is largest\n");}
    - Else {printf("C is the largest\n");}

```
} else if(b>c) {
    printf("B is the largest\n");
} else {
    printf("C is the largest\n");
}
Return 0;
}
```

Switch statement

- switch(expr) /* expr is a Boolean */
- { /* NOTE: C1 and C2 are values of type int or char only! */
  - case C1: {statement0; break; }
  - case C2: {statement1; break;}
  - ….
  - default: /* optional */
  - {default_statement; break;}
- }

Switch statement

- Char c;
- Scanf(“%c”, &c);
- switch(c) /* expr is a Boolean */
- { case ‘+’: printf(“Addition\n”); break;
- Case ‘-’: printf(“Subtraction\n”); break;
- Case ‘*’: printf(“Multiplication\n”);break;
- Case ‘/’: printf(“Division\n”); break;
- Default: printf(“Not an arithmetic operation\n”); break;
- }

Contents

- Introduction, History and Prelude
- Hello World
- Keywords and Types
- Data and Representation
- Operators and Expressions
- Decision Statement
- Loops

Loops

- while, do-while, for
- While(expr) {
  - Statement;
- }
- do { statement; } while(expr);

Loops

- for(expr1; expr2; expr3)
  - Statement;
- Semantically same as:
  - expr1;
  - while(expr2) {
    - Statement;
    - expr3;
  - }

Other ways to alter control flow

- break: terminates the execution of a loop or a switch. Resumes execution at 1st statement after loop or switch.
- continue: terminates the current iteration of the loop. Restarts execution from the beginning of the loop. If used within for loop, will increment loop counter.
- goto label: Unconditionally jumps to ‘label’. Use carefully!!