Programming Fundamentals (CS 302)

Dr. Ihsan Ullah

Lecturer
Department of Computer Science & IT
University of Balochistan

Outline

- Introduction
- Program development
- C language and beginning with programming
- Compilation steps of a C program
- Variables and their data types
- Format specifiers and escape sequences
- C operators
- Taking input
- Comments

Computer Programming

- A program is a set of ordered instructions that enables a computer to solve a problem
- The process of developing and implementing these steps is called programming
- Instructions must be provided to the computer in a systematic order

Computer Components

- Hardware
  - Physical parts of a computer
  - CPU, RAM, etc.
- Software
  - A collection of computer programs and data
  - Makes use of computer hardware
  - Guides the computer at each and every step

Types of computer software

- Application software
  - To perform specific tasks of computer users
  - Word processors, Spreadsheets, Payroll, Accounts
- System software
  - Control, operate and monitor the computer through interacting with the hardware
  - An interface between the hardware and the application software
  - Operating systems, device drivers, compilers, interpreters

Programming language

- Enables instructing the computer to perform specific tasks
- Based on rules of syntax and semantics
- Evolution
  - Efficient translation of human language
    - High-level syntax, procedural/structured languages
  - Complexity management
    - Object oriented approach

Types of languages

- Low level languages
  - Machine oriented languages that interact with the machine at low level
  - Detailed knowledge of computer hardware and its configuration
  - Machine language
    - Strings of **0's** and **1's**
    - Fast but complex and difficult to debug
  - Assembly language
    - **0's** and **1's** are replaced by mnemonic codes
    - Assembler translates the code into machine language

Types of languages

- High level languages
  - Use English-like instructions
  - Abstracts over the target hardware
  - Easy to learn and use
  - Desirable if achievable
  - **COBOL, FORTRAN, Pascal,** etc.
  - Use **compiler/interpreter** to translate high level code instructions into machine language

Program development

1. Define the problem
2. Outline the solution
3. Develop the outline into an algorithm
4. Test the algorithm for correctness
5. Code the algorithm into a specific programming language
6. Run the program on the computer
7. Document and maintain the program

Define the problem

- Input
  - What we got?
- Output
  - What we want to get?
- Processing
  - How do we get the desired output from input?

- Area of a rectangle (A=L x W)

Outline the solution

- Identification of major steps to solve the problem
- Subtasks involved
- Major variables and data structures (L,W)
- Major control structures
- The underlying logic (A=L * W)

Develop the outline into an algorithm

- An algorithm is a specification of precise and ordered steps that describe the tasks to be performed to solve a problem (pseudo code)
  - Start
  - Input L
  - Input W
  - Calculate Area, A=W * L
  - Display output, A
  - End

Test the algorithm for correctness

- Evaluating the algorithm through test data
- Comparing the obtained results to actual ones
- Correcting logic errors

Code the algorithm

- Coding the algorithm using some chosen programming language
- Language-specific considerations are made here

Run the program on the computer

- Translation of the high-level code into low level
  - Correct compile time errors
- Run the compiled code
  - Correct runtime errors (incorrect inputs)
  - Correct logic errors (incorrect results)

Document and maintain the program

- Documentation of steps involved in developing algorithm and code
- Maintenance and updating of program

Outline

- Introduction
- Program development
- C language
- Compilation steps of a C program
- Variables and their data types
- Format specifiers and escape sequences
- C operators
- Taking input
- Comments

C Language

- Dennis Ritchie, 1972, Bell Labs.
- Successor of B formerly BCPL (Basic Combined Programming Language)
- Strongly associated with UNIX
- Incorporate feathers of high level and assembly languages
- C programs are efficient and fast
- C programs are fairly portable
- C language has a simple and well-structured syntax

A simple C program

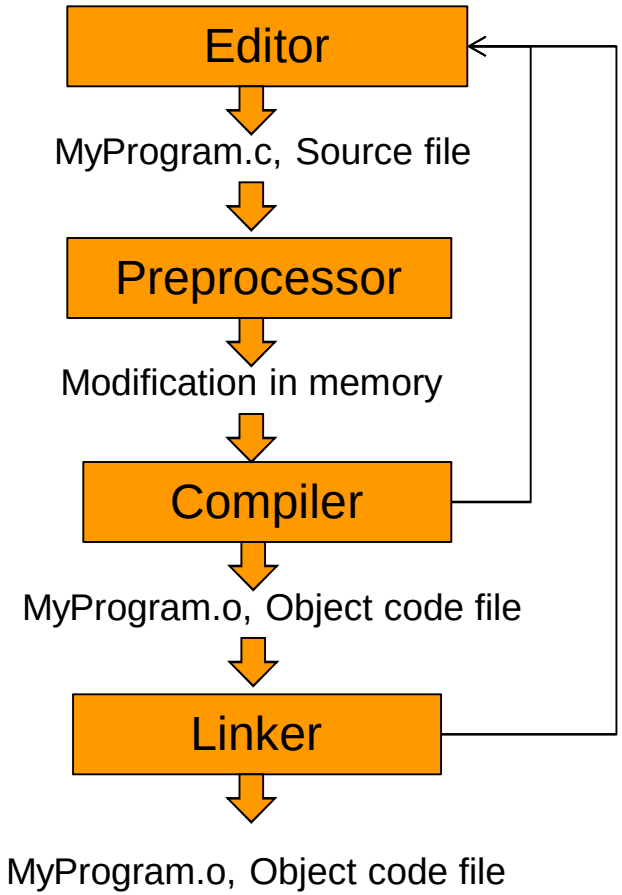
```
#include <stdio.h>
   void main(void)
   {

        printf("Welcome to C language");
   }
```

instructing the preprocessor to include header file stdio.h for the built function printf()

main function declaration. One main () function is mandatory for a program to get control from OS

Use of the built in function, printf(), to print a simple string on the screen

Compilation of a C program



Variables

- Variable is a space in memory identified through a given name
- Can represent a numerical value, character or string of characters
- Variables must be declared at the beginning of a program with their proper types
- A variable name can contain alphabets, digits and underscore
- A name cannot start with a digit
- Keywords cannot be used as variable names

Variable data types

A variable can hold a specific type of data which must be defined

Data Type	Range of Values	Space in memory
char or signed char	-128 to 127	1 byte
unsigned char	0 to 255	1 byte
int	-32768 to 32767	2 bytes
long int	-2,147483,648 to 2,147483,647	4 bytes
float	3.4e-38 to 3.4e+38	
double	1.7e-308 to 1.7e+308	

The above-given sizes are the minimum for these variables. In practice, they depend upon the implemented compiler

C Keywords

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

Format specifiers

Format specifiers are used to format the printed output

Format specifier	Variable type
%d	int
%c	char
%f	float
%lf	double
%s	string

Escape Sequences

Special characters reserved for specific tasks such as changing the line or printing some symbol which perform other tasks when used directly

Escape Code	Use
\n	New line
\”	Double quote
\t	Tab space
\\	Backslash
\b	backspace
\f	Form feed

Example: Format specifiers & escape sequences

```
# include <stdio.h>
void main(void)
    {
    int a=34;
    float b=4.61;
    printf(“The value of \”integer\” a=%d \n”,a);
    printf(“The value of \”float\” b=%f”,b);
    }
```

Arithmetic operators

- Addition, +
- Subtraction, -
- Division, /
- Multiplication, *
- Modulus (remainder), % (9%2=1)
- Increment, ++
- Decrement, --
- Example
  - A+B => A and B are operands and + is an operator

Assignment operators

- Simple assignment, = *(a=4)*
- Addition & assignment, += *(a+=5 => a=a+5)*
- Subtraction & assignment, -+
- Multiplication & assignment, *=
- Division & assignment, /=
- Modulus & assignment, %=

Relational operators

- Equal, ==
- Not-equal, !=
- Less than, <
- Greater than, >
- Less than or equal, <=
- Greater than or equal, >=

Logical operators

- And (&&) operator returns true if both of its operands are true
- Or (||) operator, returns true if at least one of its operands are true
- Not (!) operator reverses the logical state of its operand (from true to false and from false to true)

Operators' precedence

Name	Operator
Logical NOT	!
Arithmetic	* / %
Arithmetic	+-
Relational	< > <= >=
Relational	== !=
Logical AND	&&
Logical OR	||
Assignment	= += -= *= /= %=

Parenthesis are evaluated first. This order can be changed by using parenthesis

Example operators

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

Taking input

```
# include <stdio.h>
void main (void)
    {
    float length, width, area;
    printf("please enter length:");
    scanf("%f", &length);
    printf("please enter width:");
    scanf("%f",&width);
    area=length*width;
    printf("area=%f ",area);
    }
```

Lab task

- Write a program that inputs the radius of a circle and estimates its area
  - $A=Pi * r^2$

Comments

- Comments provide information about the program
- Compiler does not read comments
- Single line comments // .. comments ..
- Multiline comments /* ... comments … */

Summary

Now you know,

- What is programming and why we need programming languages?
- Steps to develop a program
- What is C language?
- Writing simple programs
  - Printing on display
  - Using variables
  - Formatting the output and using escape sequences
  - Taking input
  - Use of comments