C Basics

PROF. NAVRATI SAXENA

Contents

- Variable
- Variable names
- Data Types
- Operators
  - Arithmetic
  - Relational
  - Equality
  - Logical

High-level view of programming

- **Compile** the source code
  - **Compilation** is the process of converting the source code into **machine code**
- **Run** or **execute** the machine-language file

A Simple C Program

```
#include <stdio.h> /* a simple program */
int main()
{
        printf("Simple C program");
        return 0;
}
```

- Comments
  - Text surrounded by **`/*`** and **`*/`**
  - Ignored by computer
  - Used to describe program

- int main()
- { }

`#include <stdio.h>`

Preprocessor directive

Tells computer to load contents of a certain file

`<stdio.h>` allows standard input/output operations

Variable in C

- Variable declaration
- Variable initialization
- Main body

Variables in Programming

- Used to store/retrieve data over life of program
- Type of variable determines what can be placed
- *Assignment* – Placing a value in a variable
- Variables must be *declared* before they are assigned
- The value of a *variable* can change; A *constant* always has the same value

Naming variables

- When a variable is *declared* it is given a name

- Good programming practices
  - Choose a name that reflects the role of the variable in a program, e.g.
    - Good: customer_name, ss_number;
    - Bad : cn, ss;
  - Don't be afraid to have long names if it aids in readability

- Restrictions
  - Name must begin with a letter; otherwise, can contain digits or any other characters. C is CASE SENSITIVE! Use 31 or fewer characters to aid in portability

Variable Declaration

- All variables must be **declared** in a C program before the first executable statement!
  Examples:

```
main(){
    int a, b, c;
    float d;
    /* rest of the program, goes here */
}
```

C Variable Names

- May only consist of letters, digits, and underscores
- May not begin with a digit
- Variable names in C are case sensitive
- Should be very descriptive

Variable assignment

After variables are declared, they must (should) be given values.

This is called **assignment** and it is done with the '=' operator. Examples:

```
float a, b;
int c;
b = 2.12;
c = 200;
```

C Data Types

Basic C Data Types

- There are four basic data types in C:
  - char
    - A single byte capable of holding one character in the local character set.
  - int
    - An integer of unspecified size
  - float
    - Single-precision floating point
  - double
    - Double-precision floating point

Char variable type

- Represents a single *byte* (8 *bits*) of storage
- Can be *signed* or *unsigned*
- Internally char is just a number
- ASCII character set used in ANSI C

int variable type

- Represents a signed integer of typically 4 or 8 bytes (32 or 64 bits)
- Precise size is machine-dependent

float and double variable types

- Represent typically 32 and/or 64 bit real numbers
- How these are represented internally and their precise sizes depend on the architecture.

Additional variable types

- Note that other types can be constructed using the modifiers:
  - short, long, signed, unsigned
- The precise sizes of these types is machine-specific

Declaring variables

- All variables must always be *declared*

- Variable declarations are always:
  - var_type *var_name;*
    - int age;
    - float annual_salary;
    - double weight, height; /* multiple vars ok */

- Memory is set aside for them, but they are not meaningful until a value is assigned to them

Assigning values to Variables

- Using the “=“ operator.
- Examples

```
int age = 52; //joint declaration/assignment

double salary;

salary = 150000.23;

age = 53; //value may change at any time
```

Structure of a C program

- So far our C programs are as follows:

```
/* description of program */

#include <stdio.h>

/* any other includes go here */

int main(){

 /* program body */

return 0;

}
```

- Let's learn more about the structure of "program body"

Statements

- All *statements* end with a semicolon!
- Commas separate multiple declarations
- Blank lines have no effect
- Extra spaces between *tokens* has no effect.
- *Comments* are ignored by the compiler

Program Body – Executable Statements

- *Executable statements* always follow variable declarations/initializations

- *Executable statements* include
  - Any valid C code that is not a declaration
    - Assignment statements
    - Arithmetic statements
    - Print statements etc.

printf() examples

- Sends output to *standard out - screen*
- printf(“%s\n”, “hello world”);
  - Translated: “print hello world as a *string* followed by a newline character”
- printf(“%d\t%d\n”, j, k);
  - Translated: “print the value of the variable j as an integer followed by a tab followed by the value of the variable k as an integer followed by a new line.”
- printf(“%f : %f : %f\n”, x, y, z);
  - English: “print the value of the floating point variable x, followed by a space, then a colon, then a space, etc.

Invisible characters

- Some special characters are not visible directly in the output stream.

- These all begin with an escape character (ie \);

  - \n newline
  - \t horizontal tab

Arithmetic Operations

- Five simple *binary arithmetic operators*

1. + “plus” → c = a + b
2. - “minus” → c = a - b
3. * “times” → c = a * b
4. / “divided by” c = a/b
5. % “modulus” c = a % b

Relational Operators

- Four basic operators for comparison of values

1. > “greater than”
2. < “less than”
3. >= “greater than or equal to”
4. <= “less than or equal to”

Equality Operators

- C distinguished between relational and *equality operators*
- This is mainly to clarify rules of order of precedence
- Two equality operators
  1. == “is equal to”
  2. != “is not equal to”

Logical Operators

- To create compound expressions
- There are two logical operators in C

1. || “logical or”
   - A compound expression formed with || evaluates to 1 (true) if any one of its components is true

2. && “logical and”
   - A compound expression formed with && evaluates to true if all of its components are true

Logical Operators, cont.

Logical operators, like relational operators, are typically used in conditional expressions

1. if ((a == 1) && (b < 3) || (c == 1)) etc.

Reading keyboard input

To be useful, program must be able to read data from external source, e.g.

- User input from keyboard
- Database
- File
- Socket

scanf function

- In <stdio.h>, so no new #include(‘s)
- Basic syntax
  - scanf(format-specifier, &var1, &var2, etc.);
  - Format-specifier is identical to printf
  - We do not need to understand everything here, just enough to do some basic I/O
- Examples
  - int a; scanf(“%d”,&a);
  - double x; scanf(“%f”,&x);
- Blocks program until user enters input!

While loops

- For repeating a statement/group of statements until some specified condition is met
- General form:

```
while (expr)
{
  statement1;
  statement2;
  ……
 }
```

- If *expr* evaluates to true (i.e. not 0), then perform statement1, etc. Otherwise, skip to end of while block

Thank you!