C Programming Basics

Ritu Arora
Texas Advanced Computing Center
June 12th, 2012

Email: rauta@tacc.utexas.edu
Recap - General

• Every C program has a function main

• Use printf function to print to the screen

• Standard functions for I/O are available in stdio.h

• Comments are of two types – single line (// ) or multi-line (/* */)

• Keywords are reserved words – e.g., include, return

• Variables are information-storage places, must have a data type associated with them and must be declared before they are used

• Basic types: int, char, long, short, float, and double

• Operators
Recap: Operators

- **Arithmetic:** +, -, /, *, %, ++, --, =

- **Relational:** a == b, a != b, a > b, a < b, a >= b, a <= b

- **Logical:** !a, a && b, a || b

- **Member and Pointer:** a[], *a, &a, a->b, a.b

- **Other:** sizeof

- **Bitwise:** ~a, a&b, a|b, a^b, a<<b, a>>b

- **More about operators and precedence:**
Recap: Type Conversion

```c
#include <stdio.h>

int main(){
    double varA;
    int varB;
    double varC = 9.34;
    varB = 2;
    varA = varB;
    varB = varC;
    printf("varB: %d, varA: %lf, varC: %lf", varB, varA, varC);
    return 0;
}
```

Note
- `double` to `int` causes removal of the fractional part
- `int` to `double` conversion happened implicitly

Output:
```
varB: 9, varA: 2.000000, varC: 9.340000
```
#include <stdio.h>

int main(){
    int varA;
    float varB;
    char varC;

    varA = 40;
    varB = 3.14159265;
    varC = 'T';

    printf("varA: %d, varB: %f, varC: %c", varA, varB, varC);
    return 0;
}

Output:
varA: 40, varB: 3.141593, varC: T
Overview of the Course

- Writing a Basic C Program
- Understanding Errors
- Comments Keywords, Identifiers, Variables
- Operators
- **Standard Input** and Output (Basic)
- Control Structures
- Standard Input and Output
- Arrays, Structures
- Functions in C
- Pointers
- Working with Files

*All the concepts will be accompanied with examples.*
#include <stdio.h>

int main(){
    char myName[50];
    printf("What is your name?\n");
    fflush(stdout);
    scanf("%s", &myName);
    printf("Hello %s!\n", &myName);
    return 0;
}

**scanf** function is used to read the keyboard input

**fflush** flushes the contents of the output buffer
#include <stdio.h>
int main(){

    char myName[50];

    printf("What is your name?\n");

    fflush(stdout);

    scanf("%s", &myName);

    printf("Hello %s!\n", &myName);
    return 0;
}

This is a **variable declaration** for string type and **myName** is a string variable. It provides storage for the information you enter. Note the usage of **char**.

Explicit flushing of the output stream

Function to read the value from keyboard and store it in computer’s memory
More Information on `scanf`

- Function to read information from the keyboard
  ```c
  scanf("%s", &myName);
  ```

- First parameter is a type-specifier
  - `%s` is a type-specifier that is used if input data is string or text.
  - other type-specifiers are `%c` for character, `%d` for decimal, `%f` for float, `%o` for octal, `%x` for hexadecimal

- The second parameter is the address of the variable that would store the value being input from the keyboard
  - `myName` is the string variable for storing the input value
  - Ampersand (`&`) before the variable name helps `scanf` find the location of the string variable in memory
Pop Quiz
(Reflect on this & ask questions, if any)

- How will you use `scanf` to read different data types?

- How will you instruct the compiler to ignore certain lines of code during program compilation?

- Fill in the blanks(_): `scanf("%__", __myIntegerNumber);`
Overview of the Course

- Writing a Basic C Program
- Understanding Errors
- Comments Keywords, Identifiers, Variables
- Operators
- Standard Input and Output (Basic)
- Control Structures
- Standard Input and Output
- Arrays, Structures
- Functions in C
- Pointers
- Working with Files

All the concepts will be accompanied with examples.
Control Structures

• **Sequence Structure** is a sequence of statements

• **Selection Structure** used for branching

• **Loop Structure** used for iteration or repetition
Conditional Expressions

• Use **if–else** or ternary operator (?:)

```plaintext
if (a > b) {
    z = a;
} else {
    z = b;
}
```

\[ z = (a > b) \ ? \ a : b \ ; \ // z = \text{max} \ (a, b) \]
if-else: Logical Expressions

```c
if (temp > 75 && temp < 80) {
    printf("It’s nice weather outside\n");
}

if (value == 'e' || value == 'n') {
    printf("\nExiting the program.\n");
} else {
    printf("\nIn the program.\n");
}
```
Decision Making, Multi-Way Decisions

• Decisions are expressed by \texttt{if-else} where the \texttt{else} part is optional

\begin{verbatim}
if (expression)
    statement1
else
    statement2
\end{verbatim}

• Multi-way decisions are expressed using \texttt{else-if} statements

\begin{verbatim}
if (expression1)
    statement1
else if (expression2)
    statement2
else
    statement3
\end{verbatim}
Multi-Way Decision

- The `switch` statement is a multi-way decision
- It tests whether an expression matches one of a number of constant integer values, and branches accordingly

```java
switch (expression) {
    case const-expression1: statements1
    case const-expression2: statements2
    default: statements3
}
```
Multi-Way Decision Example 1: multiWay1.c

```c
char c;
//other code

c = getchar();
--- the character read from the keyboard is stored in variable c
if(c=='1')
    printf("Beverage\nThat will be $8.00\n");
else if(c=='2')
    printf("Candy\nThat will be $5.50\n");
else if(c=='3')
    printf("Hot dog\nThat will be $10.00\n");
else if(c=='4')
    printf("Popcorn\nThat will be $7.50\n");
else
    printf("That is not a proper selection.\n");
    printf("I’ll assume you’re just not hungry.\n");
    printf("Can I help whoever’s next?\n");
}
//This is just a code snippet. For complete program, see file multiWay1.c
```
Output of multiWay1.c

Please make your treat selection:
1 - Beverage.
2 - Candy.
3 - Hot dog.
4 - Popcorn.

3 <enter>
Your choice: Hot dog
That will be $10.00
c = getchar();
switch(c){
    case '1':
        printf("Beverage\nThat will be $8.00\n");
        break;
    case '2':
        printf("Candy\nThat will be $5.50\n");
        break;
    case '3':
        printf("Hot dog\nThat will be $10.00\n");
        break;
    case '4':
        printf("Popcorn\nThat will be $7.50\n");
        break;
    default:
        printf("That is not a proper selection.\n");
        printf("I'll assume you're just not hungry.\n");
        printf("Can I help whoever's next?\n");
}

//This is just a code snippet. For complete program, see file multiWay2.c
Loops

• For repeating a sequence of steps/statements

• The statements in a loop are executed a specific number of times, or until a certain condition is met

• Three types of loops
  – for
  – while
  – do-while
for Loop

for (start_value; end_condition; stride)
  statement;

for (start_value; end_condition; stride) {
  statement1;
  statement2;
  statement3;
}
for  Loop Example 1: forLoop.c

```c
#include <stdio.h>

int main(){
    int i;
    for(i = 0 ; i <= 10 ; i = i+2){
        printf("What a wonderful class!\n");
    }
    return 0;
}
```

Output:
- What a wonderful class!
- What a wonderful class!
- What a wonderful class!
- What a wonderful class!
- What a wonderful class!
#include <stdio.h>

int main(){
    int i, sum;
    sum = 0;
    for(i = 1 ; i <= 100 ; i = i+1){
        sum = sum + i;
    }
    printf("Sum of first 100 numbers is: %d ", sum);
    return 0;
}

Output:
Sum of first 100 numbers is: 5050

Did you notice how multiple variables can be declared in the same line?
while Loop

• The while loop can be used if you don’t know how many times a loop should run

```java
while (condition_is_true){
    statement (s);
}
```

• The statements in the loop are executed until the loop condition is true

• The condition that controls the loop can be modified inside the loop (this is true in the case of for loops too!)
#include <stdio.h>

int main(){
    int counter, value;
    value = 5;
    counter = 0;
    while ( counter < value ){
        counter++; // Equivalent to counter = counter + 1;
        printf("counter value is: %d\n", counter);
    }
    return 0;
}
**do-while Loop**

- This loop is guaranteed to execute at least once

```c
do {
    statement (s);
} while (condition_is_true);
```
### do-while Example: doWhile.c

```c
#include <stdio.h>

int main()
{
    int counter, value;
    value = 5;
    counter = 0;
    do {
        counter++;
        printf("counter value is: %d\n", counter);
    } while (counter < value);
    return 0;
}

Note the semi-colon after specifying while

Output same as that of the while loop program shown earlier
```
Keyword: **break**

- **break** is the keyword used to stop the loop in which it is present

```c
for(i = 10; i > 0; i = i-1){
    printf("%d\n",i);
    if (i < 5){
        break;
    }
}
```

Output:

```
10
9
8
7
6
5
4
```
**continue** Keyword: myContinue.c

- **continue** is used to skip the rest of the commands in the loop and start from the top again
- The loop variable must still be incremented though

```c
#include <stdio.h>
int main()
{
    int i;
    i = 0;
    while ( i < 20 )
    {
        i++;
        continue;
        printf("Nothing to see\n");
    }
    return 0;
}
```

The `printf` statement is skipped, therefore no output on screen.
Homework 3

1. Write a C program that prompts the user to enter two integers, adds the two integers, and then prints the sum of the integers to the screen.

2. Write a C program that prompts the user to enter two integers, finds the larger of the two integers, and prints it to the screen.
   – Hint: use `if-else`
References

- C Programming Language, Brian Kernighan and Dennis Ritchie
- Let Us C, Yashavant Kanetkar
- C for Dummies, Dan Gookin
- http://cplusplus.com