Most good programmers do programming not because they expect to get paid, or adulation by the public, but because it is fun to program.

Linus Torvalds

Creator of the Linux kernel
17th in Time 100: The Most Important People of the Century!

CS101: Introduction to Programming

Venkatesh Vinayakarao
Department of Computer Science and Engineering
IIIT Sri City, Chittoor.
venkatesh.v@iiits.in

Announcements

- Labs for first week are cancelled.
  - You need to do the lab assignments. You need not submit them. They are not graded.
- Labs for future weeks till we have the real labs ready “may”
  - Have only two sections – PC Sec A and PC Sec B.
  - You will use the classrooms and share your laptops in groups to do the lab work.
- Assignment 1 is released today.
  - Form groups of 5 people. Name your team. Upload assignments to google classroom.
    - Classcode for PC2018, Section A: mwj7ptw
    - Classcode for PC2018, Section B: 1lanbmm
- Be prepared for a lab exam. %Weight for lab exam will be announced soon.

Agenda

- Writing Simple Arithmetic Programs
  - Data Handling
    - Variables and Constants
    - Operators and Operator Precedence
    - Data Types
  - We Make So Many Decisions!
    - If-else
    - Conditional Operators
    - Switch Case
  - Life Goes Round and Round!
    - For, While and Do Loops
    - Break and Continue

How does our Brain Work?

Process

Standard Knowledge

x + y = 10 + 20 = 30

Input

x = 10, y = 20

Output

30

Task: Add two given numbers

Calculating Simple Interest Again

```c
#include<stdio.h>

int main() {
    int amount, rate, time, si;

    printf("\nEnter Principal: ");
    scanf("%d", &amount);

    printf("\nEnter Rate of Interest: ");
    scanf("%d", &rate);

    printf("\nEnter Time: ");
    scanf("%d", &time);

    si = (amount * rate * time) / 100;
    printf("\nSimple Interest : %d", si);

    return(0);
}
```

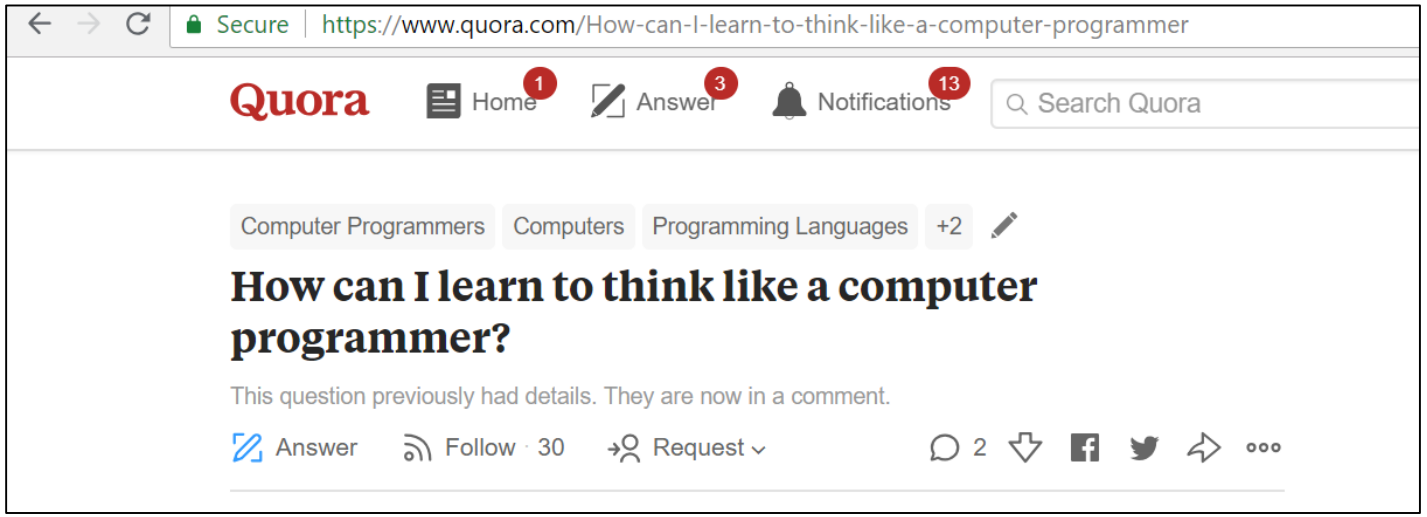
Terminology

- int is a Data Type
- main() is a Function
- amount, rate, time and si are Variables

A Fun Exercise (0 Credits)

Deadline: 6th Sep 2018.

- Read https://www.quora.com/How-can-I-learn-to-think-like-a-computer-programmer



- Make a 1 to 2 minute video on this topic. Best work will reach my colleagues in academia and industry. You may work alone or in groups of any size.

Adding Two Numbers

```c
#include<stdio.h>

int main() {
    int a=10, b=20, sum;

    sum = a + b;

    printf("Sum = %d", sum);

    return(0);
}
```

Quiz

Which of the following is true?

1. a, b and sum are variables.
2. + and = are operators.
3. int is a datatype.
4. All of the above. ✓

What is the difference?

```c
#include<stdio.h>

int main() {
    int a=10, b=20, sum;

    sum = a + b;

    printf("Sum = %d", sum);

    return(0);
}
```

Works!

```c
#include<stdio.h>

int Main() {
    int a=10, b=20, sum;

    sum = a + b;

    printf("Sum = %d", sum);

    return(0);
}
```

Fails!

Variable Names

- C is case-sensitive
  - Main and main are not the same!
  - Int A and int a are different.

This works! But, writing code like this is considered bad. As a convention, use lowercase variable names.

```c
#include<stdio.h>

int main() {
    int a=10, A=21, sum;

    sum = a + A;

    printf("Sum = %d", sum);

    return(0);
}
```

Rules for Variable Names

- Max length = 32
- Alphabets, digits or underscores are allowed
- Cannot start with a digit

Quiz: Which of the following are valid variable names?

1. $hello
2. height in feet
3. 10.or.E
4. salary ✓

Do not memorize.
Follow the naming convention.

Operator Precedence

```c
#include<stdio.h>

int main() {
    int a=10, b=5, c=2, result;

    result = a + b * c;

    printf("Result = %d", result);

    return(0);
}
```

Quiz: What will be printed?

1. 20
2. 30

*** Has higher precedence than +. Therefore, a+b*c becomes a + (b*c).**

C Operator Precedence Table

Operator	Description	Associativity
( ) [ ] . -> ++ --	Parentheses (function call) (see Note 1) Brackets (array subscript) Member selection via object name Member selection via pointer Postfix increment/decrement (see Note 2)	left-to-right
++ -- + - ! ~ (*type*) * & sizeof	Prefix increment/decrement Unary plus/minus Logical negation/bitwise complement Cast (convert value to temporary value of *type*) Dereference Address (of operand) Determine size in bytes on this implementation	right-to-left
* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right
<< >>	Bitwise shift left, Bitwise shift right	left-to-right
< <= > >=	Relational less than/less than or equal to Relational greater than/greater than or equal to	left-to-right
== !=	Relational is equal to/is not equal to	left-to-right
&	Bitwise AND	left-to-right
^	Bitwise exclusive OR	left-to-right
|	Bitwise inclusive OR	left-to-right
&&	Logical AND	left-to-right
||	Logical OR	left-to-right
? :	Ternary conditional	right-to-left
= += -= *= /= %= &= ^= |= <<= >>=	Assignment Addition/subtraction assignment Multiplication/division assignment Modulus/bitwise AND assignment Bitwise exclusive/inclusive OR assignment Bitwise shift left/right assignment	right-to-left
,	Comma (separate expressions)	left-to-right

*We will learn many of these operators later in this course. Do not memorize this table. It will be given to you in exam.

How to Use the Table?

* / %	Multiplication/division/modulus	left-to-right
+ -	Addition/subtraction	left-to-right

- * / % are on the same row. Implies that they are of same precedence. They are evaluated from left to right.
- * / % carry higher precedence compared to + and -.

Quiz: What is 24 / 3 * 2?

1. 16 ✓
2. 4

Quiz: What is 4 + 24 / 2?

1. 16 ✓
2. 14

Associativity

- a = b = c is (a = (b = c)) [Right to left associativity]
- a + b + c is ((a + b) + c) [Left to right associativity]
- Quiz: What is the output of

```
int main()
{
    printf("%d", 4 * 4 - 3
    return 0;
}
```

```
x = 4 * 4 - 3 * 2 + 4 / 2
  = (4 * 4) - 3 * 2 + 4 / 2
  = 16 - 3 * 2 + 4 / 2
  = 16 - (3 * 2) + 4 / 2
  = 16 - 6 + 4 / 2
  = 16 - 6 + (4 / 2)
  = 16 - 6 + 2
  = (16 - 6) + 2
  = 10 + 2
  = (10 + 2)
  = 12
```

Keep It Simple Principle

- Write simple code.
- Do not write, a * b + c * d + e - f.
- Instead write, (a * b) + (c * d) + (e - f).

Paranthesis has the highest priority.
So, everything in parenthesis gets evaluated first.

Data Types

- We used “int” type of variables.
- Most basic building blocks for data types are called Primitive Data Types.
  - int
  - float
  - char
- Let us learn them through examples.

How to Divide One by Two?

```
1   #include<stdio.h>
2
3   int main() {
4
5       int a=1, b=2, result;
6       result = a/b;
7       printf("Result = %d", result);
8
9       return(0);
10  }
11
```

Quiz: What is the result?

1. 1 ✓
2. 0.5

Float

```
1   #include<stdio.h>
2
3   int main() {
4
5       float a=1, b=2, result;
6       result = a/b;
7       printf("Result = %f", result);
8
9       return(0);
10  }
11
```

Quiz: What is the result?

1. 1
2. 0.5 ✓

Strings are an Array of Characters

```
1  #include <stdio.h>
2
3  int main(){
4      char institute[] =  "iiit";
5      printf("\"%s\"\n", institute);
6      return 0;
7  }
8
```

Note: We use %s for string, %f for float, and %d for int.

IIIT without Sri City? No Way!

```c
#include <stdio.h>

int main(){
    char institute[] =  "IIIT";
    char city[] = "Sri City";

    printf("%s %s\n", institute, city);
    return 0;
}

```

Data Types

Type	Explanation	Format Specifier
char	Smallest addressable unit of the machine that can contain basic character set. It is an integer type. Actual type can be either signed or unsigned. It contains `CHAR_BIT` bits.[3]	%c
signed char	Of the same size as `char`, but guaranteed to be signed. Capable of containing **at least** the [−127, +127] range;[3][4]	%c (or %hhi for numerical output)
unsigned char	Of the same size as `char`, but guaranteed to be unsigned. Contains **at least** the [0, 255] range.[5]	%c (or %hhu for numerical output)
short short int signed short signed short int	*Short* signed integer type. Capable of containing **at least** the [−32,767, +32,767] range;[3][4] thus, it is at least 16 bits in size. The negative value is −32767 (not −32768) due to the one's-complement and sign-magnitude representations allowed by the standard, though the two's-complement representation is much more common.[6]	%hi
unsigned short unsigned short int	*Short* unsigned integer type. Contains **at least** the [0, 65,535] range;[3][4]	%hu
int signed signed int	Basic signed integer type. Capable of containing **at least** the [−32,767, +32,767] range;[3][4] thus, it is at least 16 bits in size.	%i or %d
unsigned unsigned int	Basic unsigned integer type. Contains **at least** the [0, 65,535] range;[3][4]	%u
long long int signed long signed long int	*Long* signed integer type. Capable of containing **at least** the [−2,147,483,647, +2,147,483,647] range;[3][4] thus, it is at least 32 bits in size.	%li
unsigned long unsigned long int	*Long* unsigned integer type. Capable of containing **at least** the [0, 4,294,967,295] range;[3][4]	%lu
long long long long int	*Long long* signed integer type. Capable of containing **at least** the [−9,223,372,036,854,775,807, +9,223,372,036,854,775,807] range;[3][4] thus, it is at least 64 bits in size. Specified since the	%lli

Image source: Wikipedia

Do not memorize this list.

So, what did we discuss?

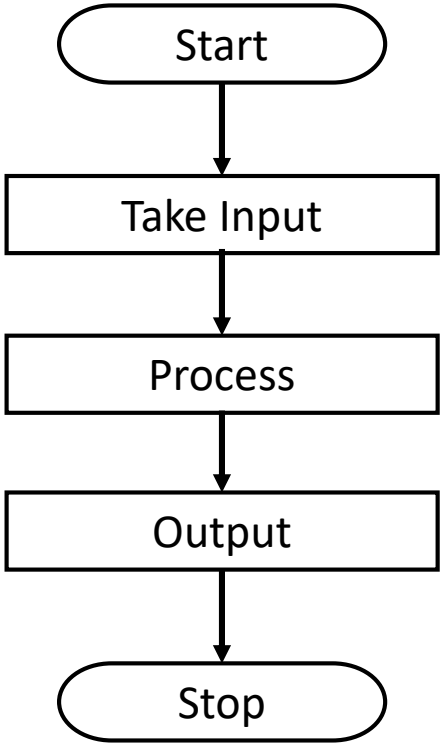
- Variables
- Data Types
- Operators & Precedence

Questions?

Decision Flow

Decisions

- Simple Programs with Linear Flow



How to handle branching?

Odd or Even?

```c
#include <stdio.h>
int main()
{
    int number;

    printf("Enter an integer: ");
    scanf("%d", &number);

    if(number % 2 == 0)
        printf("%d is even.", number);
    else
        printf("%d is odd.", number);

    return 0;
}
```

% is “modulus” operator. Returns the reminder.

We use if – else keywords to make decisions.

Positive or Negative

```c
#include <stdio.h>
int main()
{
    int number;

    printf("Enter a number: ");
    scanf("%d", &number);

    if (number <= 0)
    {
        if (number == 0)
            printf("You entered 0.");
        else
            printf("You entered a negative number.");
    }
    else
        printf("You entered a positive number.");
    return 0;
}
```

Vowel or Consonant?

```c
#include <stdio.h>
int main()
{
    char c;
    int isVowel;

    printf("Enter an alphabet: ");
    scanf("%c",&c);

    isVowel = (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u');

    if (isVowel)
        printf("%c is a vowel.", c);
    else
        printf("%c is a consonant.", c);
    return 0;
}
```

Here, we use the || operator to chain multiple conditions. Read it as "OR".

Note, = is assignment. But, == is a conditional operator.

What Went Wrong?

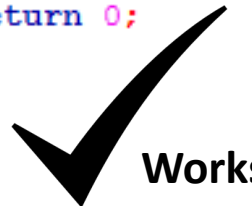
```c
#include <stdio.h>
int main()
{
    char c;
    int hasTypedI;

    printf("Enter an alphabet: ");
    scanf("%c",&c);

    hasTypedI = (c == 'I');

    if (hasTypedI)
        printf("Do say I, say us.");

    return 0;
}
```



Works!

```c
#include <stdio.h>
int main()
{
    char c;
    int hasTypedI;

    printf("Enter an alphabet: ");
    scanf("%c",&c);

    hasTypedI = (c = 'I');

    if (hasTypedI)
        printf("Do say I, say us.");

    return 0;
}
```

Fails!

Switch Case

```c
#include <stdio.h>
int main()
{
    int x = 2;
    switch (x)
    {
        case 1: printf("Choice is 1");
                break;
        case 2: printf("Choice is 2");
                break;
        case 3: printf("Choice is 3");
                break;
        default: printf("Choice other than 1, 2 and 3");
                 break;
    }
    return 0;
}
```

This is a way to avoid multiple if-else statements.

What is the Output?

```c
#include <stdio.h>
int main()
{
    int x = 2;
    switch (x)
    {
        case 1: printf("1");
        case 2: printf("2");
        case 3: printf("3");
        default: printf("X");
    }
    return 0;
}
```

Quiz: What is the output?

1. 1
2. 2
3. 123X
4. 23X ✓
5. X

So, what did we discuss?

- If-else
- Conditional Operators
- Switch Case

Questions?

Computational Thinking

Computational Thinking

- Can you swap the water in G1 with water in G2? You can use the empty glass if you wish.

A Glass of Water, G1

Empty Glass

A Glass of Water, G2

C Program to Swap Two Numbers

```
1   #include<stdio.h>
2
3   int main()
4   {
5       int a=1,b=2,c=0;
6       c=a;
7       a=b;
8       b=c;
9       printf("After swapping a=: %d b=: %d",a,b);
10  }
11
```

Can you do this without using the third variable?

Swapping Without a Third Variable

```c
#include<stdio.h>

int main()
{
    int a=1,b=2;
    a=a+b;
    b=a-b;
    a=a-b;
    printf("After swapping a=%d b=%d",a,b);
}
```

This makes coding an art!