CS1101: Foundations of Programming

Introduction and Fundamentals of C

Dept. of Computer Science & Engineering

Indian Institute of Technology Patna

General Information

- **Instructors**
  - **Jimson Mathew**
  - **Raju halder**
  - **Arijit Mondal**

- **Course webpage:** `www.iitp.ac.in/~arijit/`**, then follow** `Teaching`

Course syllabus

- Introduction to C
- Variables, data type
- Statement, Conditional statement
- Loop construct
- Array, structure, union
- Function, Recursion
- Pointers
- Stack, queue, tree
- Searching, Sorting
- File handling

Books

- Programming with C by Byron Gottfried, Schaum's Outlines Series
- The C Programming Language by Brian W Kernighan, Dennis M Ritchie
- Data structures by S. Lipschutz, Schaum's Outline Series
- C: How to Program by Paul Deitel and Harvey Deitel

Evaluation policy

- This is a 3-0-3-4.5 course
- For theory:
  - Assignment — 20%
  - Midsem — 30%
  - Endsem — 50%
- Overall
  - Weightage for theory – 2 (67%)
  - Weightage for lab – 1 (33%)

Introduction: Overview of computing systems

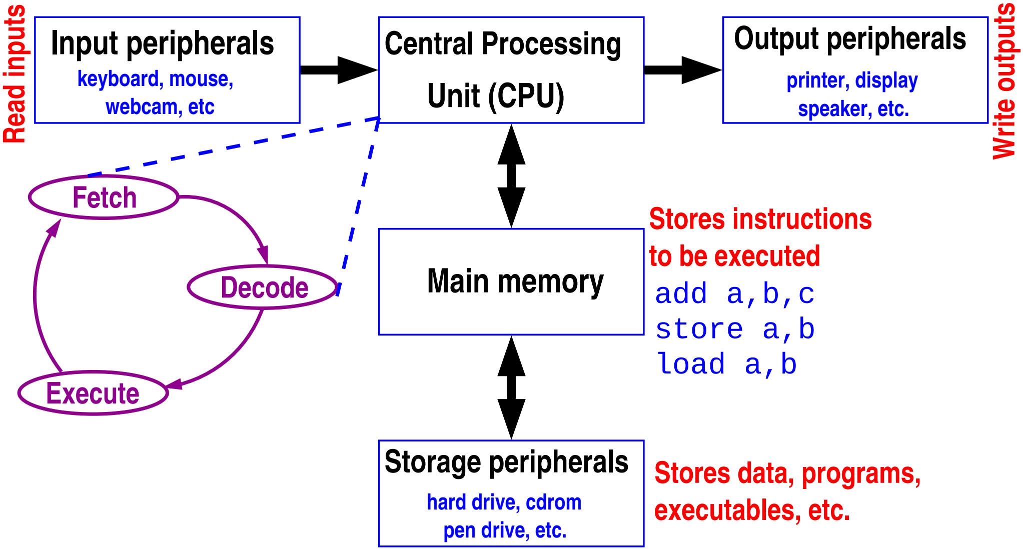
Why do we use computers?

- To solve problems with the help of machines, implementing automation for monotonous jobs, etc.
  - Find the maximum of 5 integers
  - Find the maximum of 5,000,000 integers
- Second scenario needs a systematic method to find the maximum element
- In this course, we will learn how to write programs
  - Program - set of instructions written for a given purpose. It tells computer what to do
  - *Computers are good in obeying instructions but have no intelligence!*

What can a computer do

- Check prime number
- Palindrome recognizer
- Find shortest path between two points
- Telephone pole placement
- Spaceship control
- Finger-print recognition
- Play chess
- Image recognition
- Speech recognition
- Language recognition and many more!

Simple view of computer



Read inputs

Input peripherals
keyboard, mouse,
webcam, etc

Central Processing Unit (CPU)

Output peripherals
printer, display
speaker, etc.

Write outputs

Fetch

Decode

Execute

Main memory

Stores instructions
to be executed

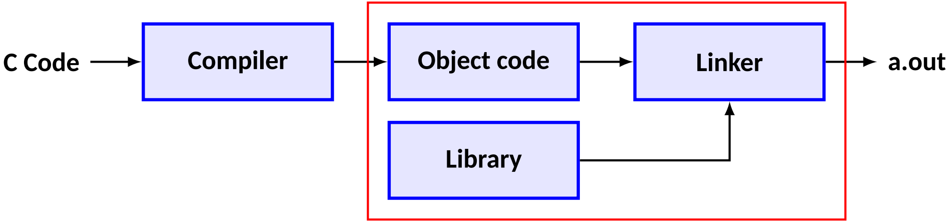
```
add a,b,c
store a,b
load a,b
```

Storage peripherals
hard drive, cdrom
pen drive, etc.

Stores data, programs,
executables, etc.

Overview of compilation

- Write the program using high-level language (C in our case)
- Compile the program (primarily `gcc` will be used) - it generates the binary (executable) code
- Run the executable code (`a.out` in our case)



Binary representation

- Normal number system uses decimal representation 0-9 which has base 10
  - **Example:** $625 = 5 \times 10^0 + 2 \times 10^1 + 6 \times 10^2$
- Numbers in computer systems are represented in base-2, it has only two digits 0 and 1
  - **Example:** $1011 = 1 \times 2^0 + 1 \times 2^1 + 0 \times 2^2 + 1 \times 2^3 = 11$
- Octal representation (base-8) - 0-7 are used
- Hexadecimal representation (base-16) - 0-9, A, B, C, D, E, F are used
  - A=10, ...,F=15
  - Conversion from binary to hexadecimal or vice-versa are easy

Bits and Bytes

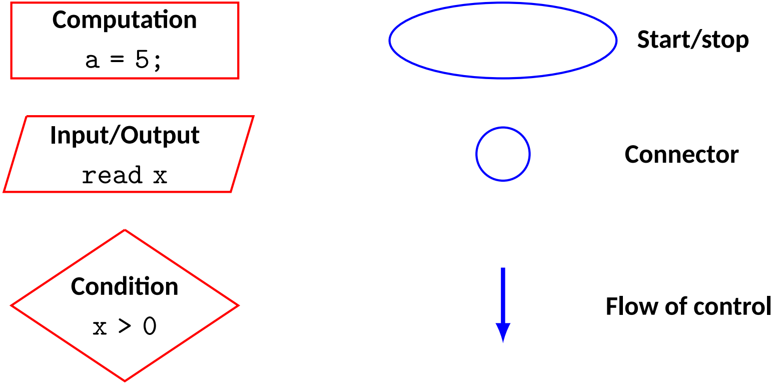
- Bit — a single binary digit that is either 0 or 1
- Byte — a consecutive sequence of 8 bits
  - 4 bytes = 32 bits
  - 8 bytes = 64 bits
- Range of integers that can be expressed depends on the representation size
  - If 1 byte is used then the range of integers will be 0-255
  - If 4 bytes are used then the range of integers will be 0-$(2^{32} - 1)$
- Different datatypes (integers, float, double) can have different sizes

Problem solving flow

- Describe the problem to be solved clearly. Specify inputs and outputs unambiguously
- Draw a flowchart of steps (or develop an algorithm) to be followed
- Convert the flowchart into a program, choose your preferred language (C in our case)
- Compile the program and generate an executable code
- Run the executable code
- Test your code / program with different inputs

Flow chart

- Representing the steps in a structured way and presenting the flow of execution of those steps
- Uses very simple basic blocks to describe the overflow of steps / algorithms



Computation

a = 5;

Input/Output

read x

Condition

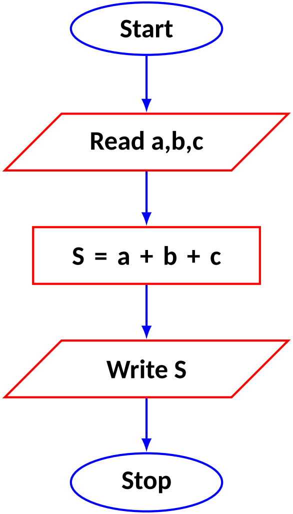
x > 0

Start/stop

Connector

Flow of control

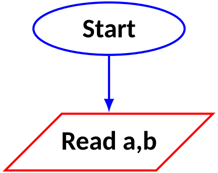
Flow diagram: Sum of 3 numbers



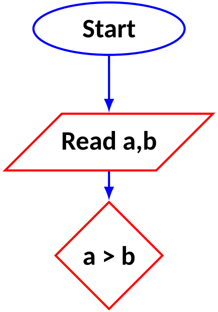
Flow diagram: Max of two numbers

Start

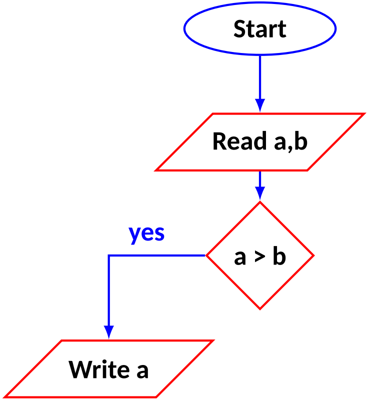
Flow diagram: Max of two numbers



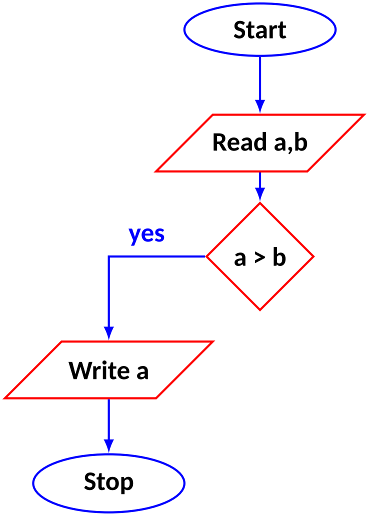
Flow diagram: Max of two numbers



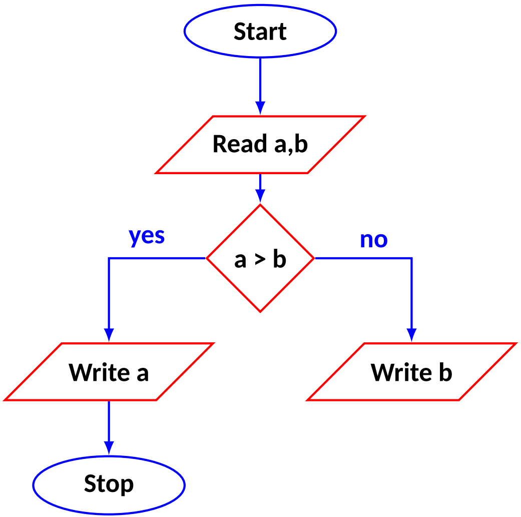
Flow diagram: Max of two numbers



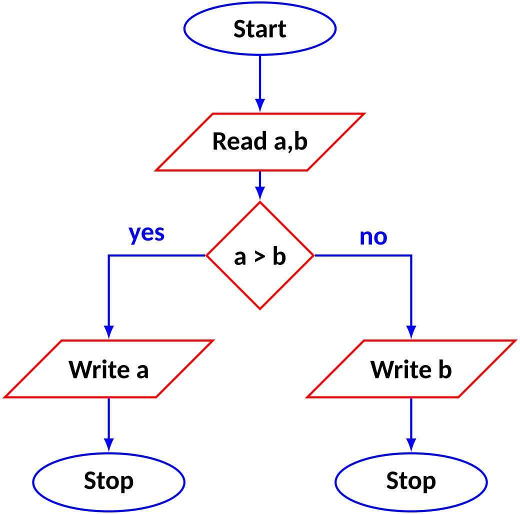
Flow diagram: Max of two numbers



Flow diagram: Max of two numbers



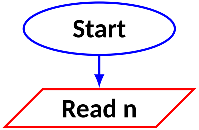
Flow diagram: Max of two numbers



Flow diagram: factorial of a number

Start

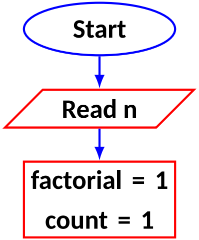
Flow diagram: factorial of a number



Start

Read n

Flow diagram: factorial of a number



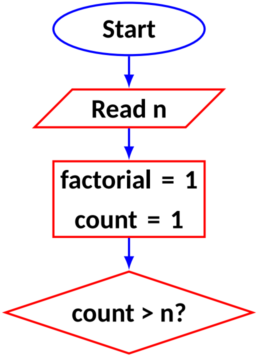
Start

Read n

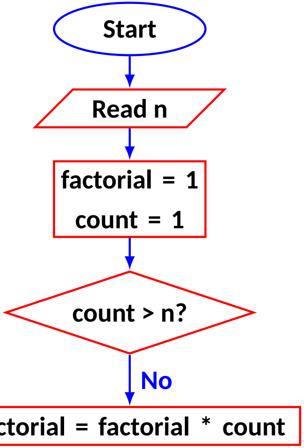
factorial = 1

count = 1

Flow diagram: factorial of a number



Flow diagram: factorial of a number


Start

Read n

factorial = 1

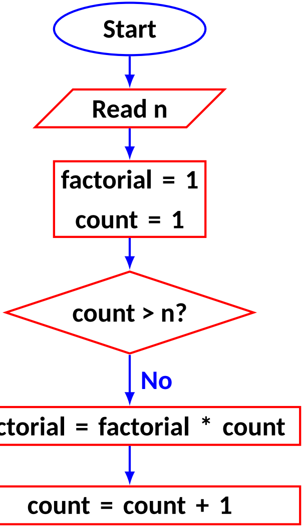
count = 1

count > n?

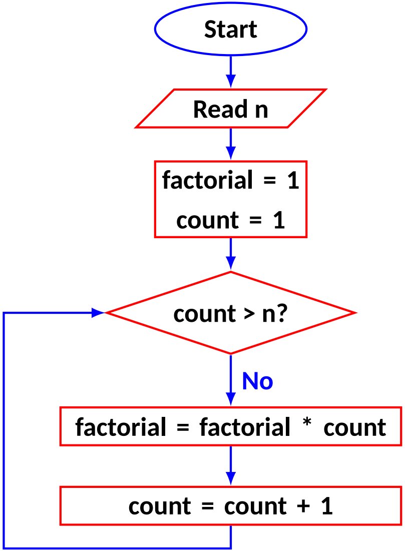
No

factorial = factorial * count

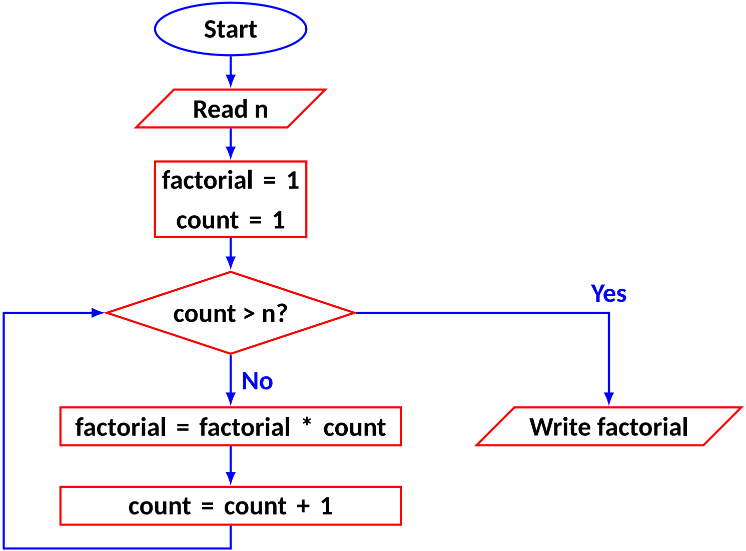
Flow diagram: factorial of a number



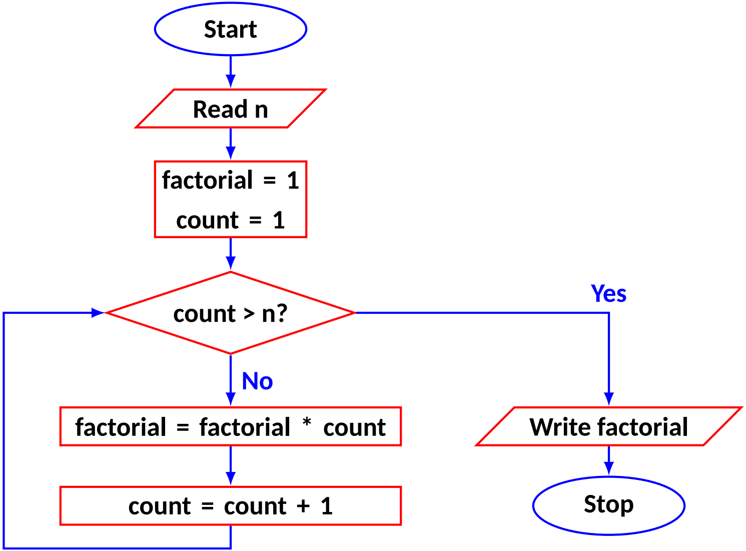
Flow diagram: factorial of a number



Flow diagram: factorial of a number



Flow diagram: factorial of a number



Fundamentals of C

First C program

```c
#include <stdio.h>
int main()
{
  printf("Hello, World!\n");
  return 0;
}
```

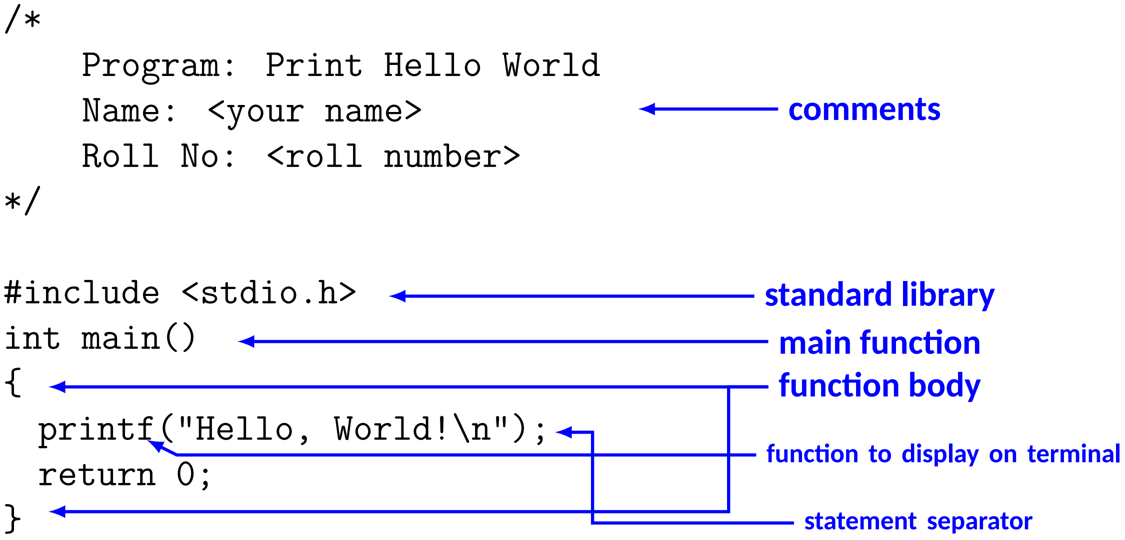
First C program

```
#include <stdio.h>
int main()
{
  printf("Hello, World!\n");
  return 0;
}
```

- **Output:**

```
Hello, World!
```

Good practice for program writing



```
/*
    Program: Print Hello World
    Name: <your name>
    Roll No: <roll number>
*/

#include <stdio.h>
int main()
{
  printf("Hello, World!\n");
  return 0;
}
```

comments

standard library

main function

function body

function to display on terminal

statement separator

C program: sum and max of two numbers

```c
#include <stdio.h>
int main(){
  int x, y, sum, max;
  scanf("%d%d",&x,&y);
  sum = x + y;
  if( x > y ){
    max = x;
  }else{
    max = y;
  }
  printf("sum = %d, max = %d \n", sum, max);
  return 0;
}
```

Structure of C program

- A collection of functions
- Exactly one special function namely main() must be there. Execution will start from this function
- Each function has different types statements
- Statements are executed one by one

C program: you need to know

- Variables
- Constants
- Expressions (Arithmetic, Logical, Assignment)
- Statements (Declaration, Assignment, Control (conditional / branching, looping))
- Arrays
- Functions
- Structures
- Pointers

The C character set

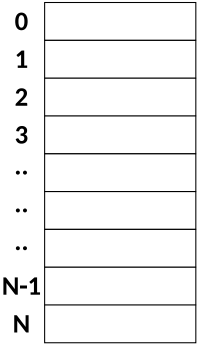
- C language alphabet
  - Uppercase letters ’A’ to ’Z’
  - Lowercase letters ’a’ to ’z’
  - Digits ’0’ to ’9’
  - Special characters: ! # % ^ & * - _ + = ~ [ ] \ | ; : ' " { } , . < > / ? and ‘blank’ (tab, space)
- A C program should not contain anything else

Variables

- Very important concept for programming
- An entity that has a value and is known to the program by a name
- Can store any temporary result while executing a program
- Can have only one value assigned to it at any given time during the execution of the program
- The value of a variable can be changed during the execution of the program
- Variables stored in memory
- Remember that memory is a list of storage locations, each having a unique address

Variables (contd.)

- A variable is like a bin
  - The contents of the bin is the value of the variable
  - The variable name is used to refer to the value of the variable
  - A variable is mapped to a location of the memory, called its address



Example

```
#include <stdio.h>
int main()
{
  int x;
  int y;
  x=1;
  y=3;
  printf("x=%d, y=%d\n",x,y);
  return 0;
}
```

= – does not denote equality

Values are assigned to variables.

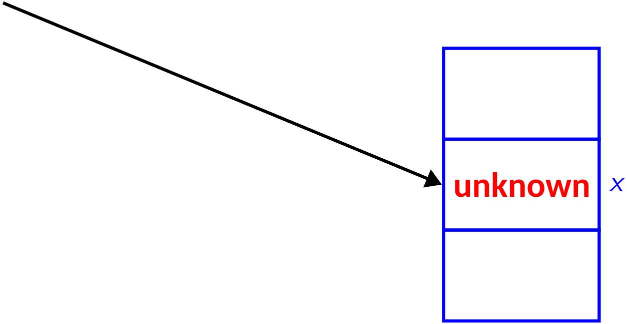
Variables in memory

$x = 10$

$x = 20$

$x = x + 1$

$x = x * 5$



unknown

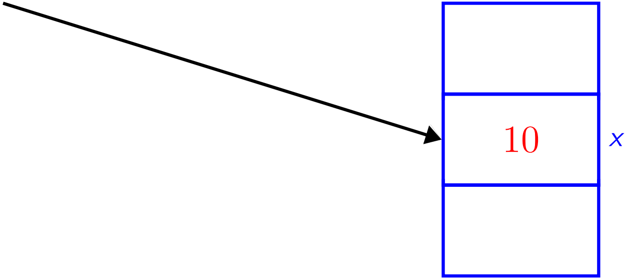
Variables in memory

$x = 10$

$x = 20$

$x = x + 1$

$x = x * 5$



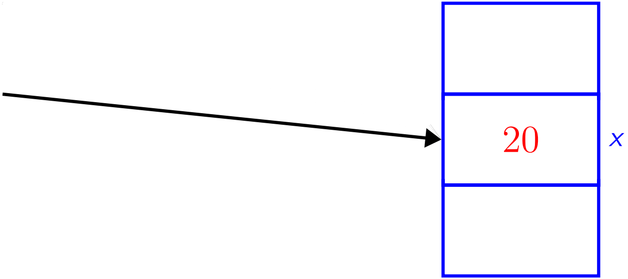
Variables in memory

$x = 10$

$x = 20$

$x = x + 1$

$x = x * 5$



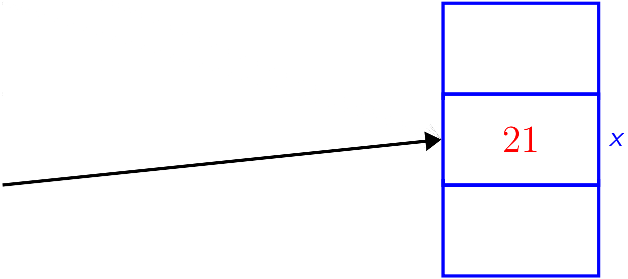
Variables in memory

$x = 10$

$x = 20$

$x = x + 1$

$x = x * 5$



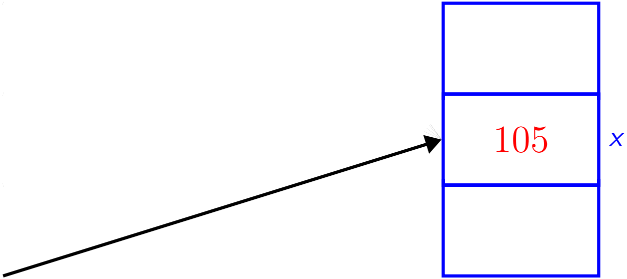
Variables in memory

$x = 10$

$x = 20$

$x = x + 1$

$x = x * 5$



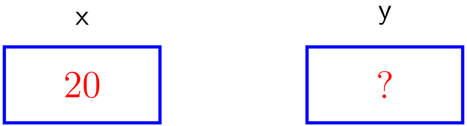
Variables in memory

$$x = 20$$

$$y = 15$$

$$x = y + 3$$

$$y = x/6$$



x

20

y

$?$

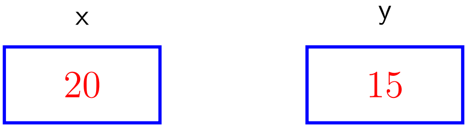
Variables in memory

$x = 20$

$\boldsymbol{y = 15}$

$x = y + 3$

$y = x/6$



x

20

y

15

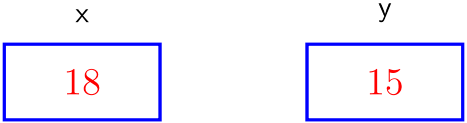
Variables in memory

$$x = 20$$

$$y = 15$$

$$x = y + 3$$

$$y = x/6$$



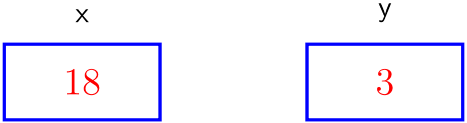
Variables in memory

$x = 20$

$y = 15$

$x = y + 3$

$y = x/6$



Data types

- Each variable has a type, indicates what type of values the variable can hold
- Four common data types in C
  - `int` - can store integers (usually 4 bytes)
  - `float` - can store single-precision floating point numbers (usually 4 bytes)
  - `double` - can store double-precision floating point numbers (usually 8 bytes)
  - `char` - can store a character (1 byte)

Data types

- Must declare a variable (specify its type and name) before using it anywhere in your program
- All variable declarations should be at the beginning of the main() or other functions
  - There are exception too
- A value can also be assigned to a variable at the time the variable is declared.
  - `int speed = 30;`
  - `char flag = 'y';`

Variable names

- Sequence of letters and digits
- First character must be a letter or '_'
- No special characters other than '_'
- No blank in between
- Names are case-sensitive (max and Max are two different names)
- Examples of valid names:
  - `i rank1 _MAX max Min class_rank`
- Examples of invalid names:
  - `a's  fact rec  2sqroot  class,rank`

Variable names

- **Valid identifiers**
  - X
  - abc
  - simple_interest
  - a123
  - LIST
  - stud_name
  - Empl_1
  - Empl_2
  - avg_empl_salary

- **Invalid identifiers**
  - 10abc
  - my-name
  - "hello"
  - simple interest
  - (area)
  - %rate

C Keywords

- Used by the C language, cannot be used as variable names
- Examples:
  - `int, float, char, double, main, if else, for, while, do, struct, union, typedef, enum, void, return, signed, unsigned, case, break, sizeof,....`
  - There are others, see textbook...

Example 1

```
#include <stdio.h>
int main(){
  int x, y, sum;
  scanf("%d%d",&x, &y);
  sum = x + y;
  printf("Summation of x=%d and y=%d is %d\n",x,y,sum);
  return 0;
}
```

variable declaration

read variables and assign

computation and assign

Example 1

```
#include <stdio.h>
int main(){
  int x, y, sum;
  scanf("%d%d",&x, &y);
  sum = x + y;
  printf("Summation of x=%d and y=%d is %d\n",x,y,sum);
  return 0;
}
```

variable declaration

read variables and assign

computation and assign

Output:

```
25 144
Summation of x=25 and y=144 is 169
```

Example 2

```
#include <stdio.h>
int main(){
  float x,y;     /* two real numbers */
  int d1=24,d2;   /* integer d1 initialized to 24 */
  scanf("%f%f%d",&x,&y,&d2);
  printf("Summation of x=%f and y=%f is %f\n",x,y,x+y);
  printf("%d minus %d is %d\n",d1,d2,d1-d2);
  return 0;
}
```

Input: `scanf` function

- Performs input from keyboard
- It requires a format string and a list of variables into which the value received from the keyboard will be stored
- format string = individual groups of characters (usually ’%’ sign, followed by a conversion character), with one character group for each variable in the list

```
int a,b;
float c;
scanf("%d%d%f",&a,&b,&c);
```

format string

Conversion characters:

c — for char

d — for int

f — for float

lf — for double

Input: `scanf` function (contd.)

- **Examples**
  - `scanf("%d", &size);`
    - **reads one integer and stores it in variable named size**
  - `scanf("%c", &nextchar);`
    - **reads one character and stores in char variable**
  - `scanf("%f", &temperature);`
    - **reads a floating (real) number**
  - `scanf("%lf", &length);`
    - **reads a double (real) number**
  - `scanf("%d%d", &x, &y);`
    - **reads in two integers**

Input: `scanf` function (contd.)

- `scanf()` **will wait for you to type the input from keyboard**
- **You must type the same number of inputs as the number of %'s in the format string**
- **Example:** `scanf("%d%d%d",...)` **this expects 3 integers to be typed from keyboard. Execution will not proceed unless it receives three inputs**

Reading a single character

- A single character can be read using `scanf` with `%c`
- It can also be read using the `getchar()` function

```
char c;
c=getchar();
```

- Program waits at the `getchar()` line until a character is typed, and then reads it and stores it in `c`

Output: `printf` function

- Performs output to the standard output device (typically defined to be the screen). It requires a format string in which we can specify:
  - The text to be printed out
  - Specifications on how to print the values

    ```
    printf("The number is %d\n", num);
    ```
  - The format specification `%d` causes the value listed after the format string to be embedded in the output as a decimal number in place of `%d`
  - Output will appear as: `The number is 25`

Output: `printf` function

- **General syntax:**

  ```
  printf (format string, arg1, arg2, ..., argn);
  ```

  - `format string` **refers to a string containing formatting information and data types of the arguments to be output**
  - **the arguments** `arg1, arg2, ...` **represent list of variables/expressions whose values are to be printed**
- **The conversion characters are the same as in** `scanf`

Examples of `printf`

```
printf("Average of %d and %d is %f", a, b, avg);
printf("Hello! \n Good Afternoon\n");
printf("%3d %5d %7d", a, b, a*a+b*b);
printf("%7.2f %5.1f", a, b);
```

Many more options are available for both `printf` and `scanf`. Check books.

Read only variable

- Variables whose values can be initialized during declaration, but cannot be changed after that
- Declared by putting the `const` keyword in front of the declaration
- Storage allocated just like any variable
- Used for variables whose values need not be changed
  - Prevents accidental change of the value

Read only variable

- **Correct**

```
int main(){
  const int Limit = 10;
  int n;
  scanf("%d",&n);
  if(n>Limit)
    printf("Out of limit\n");
  return 0;
}
```

- **Incorrect**

```
int main(){
  const int Limit = 10;
  int n;
  scanf("%d",&n);
  Limit = Limit + n;
  printf("New Limit=%d\n",Limit);
  return 0;
}
```

Constants

- Integer constants
  - Consists of a sequence of digits, with possibly a plus or a minus sign before it
  - Embedded spaces, commas and non-digit characters are not permitted between digits
- Floating point constants
  - Two different notations
  - Decimal notation: 25.0, 0.0034, .84, -2.234
  - Exponential (scientific) notation 3.45e23, 0.123e-12, 123e2
    - e means ”10 to the power of”

Constants

- Character constants
  - Contains a single character enclosed within a pair of single quote marks
  - Examples :: `'2'`, `'+'`, `'Z'`
- Some special backslash characters
  - `\n` — new line
  - `\t` — horizontal tab
  - `\'` — single quote
  - `\"` — double quote
  - `\\` — backslash
  - `\0` — null

Example-1

```
#include <stdio.h>
int main()
{
  printf("Hello, World!\n");
  printf("Hello,\n World!\n");
  return 0;
}
```

Example-2

```
#include <stdio.h>
int main()
{
  printf("Hello, World!\n");
  printf("Hello,\n World!\n");
  printf("Hello,\t World!\n");
  return 0;
}
```

Example-3

```
#include <stdio.h>
int main()
{
  int number;
  scanf("%d",&number);
  printf("Student count in this class is %d\n",number);
  return 0;
}
```

Example-4: Centigrade to Fahrenheit

```
#include <stdio.h>
int main()
{
  float cent,fahr;
  scanf("%f",&cent);
  fahr=cent*(9.0/5.0)+32;
  printf("%f C equals to %f\n",cent,fahr);
  return 0;
}
```

Example-5: Maximum of two numbers

```
#include <stdio.h>
int main()
{
  int x,y;
  scanf("%d%d",&x,&y);
  if(x>y){ printf("Largest is %d\n",x);}
  else{ printf("Largest is %d\n",y);}
  return 0;
}
```

Example-6: What will be the output?

```
#include <stdio.h>
int main()
{
  int x,y;
  scanf("%d%d",&x,&y);
  if(x>y){ printf("Largest is %d\n",x);}
  printf("Largest is %d\n",y);
  return 0;
}
```

Linux commands

- `ls` — **Lists all files in a directory. Try** — `ls`, `ls -l`, `ls -al`, `ls -lrt`
- `cat` — `cat <filename>` **- displays the content of the file**
- `cd` — `cd <dirname>` **- change directory**
- `cp` — `cp <src> <dest>` **- copies file**
- `mv` — `mv <src> <dest>` **- renaming a file**
- `pwd` — **print present working directory**
- `mkdir` — `mkdir <dirname>` **- create a new directory**
- `rm` — `rm <filename>` **- remove / delete a file. Deleted file cannot be recovered**
- `man` — `man <help-topic>` **- manual page**

Practice problems

- Read two integers and two double numbers, each in separate `scanf()` statement and print them in a single `printf()` statement.
- Repeat above for `float` and `char` data types
- Read two integers and print them in separate lines such that the last digit of each integer is exactly 8 spaces away from the beginning of the line it is printed in