Introduction to Computer & Lab

Lecture No. 1

Course Overview

- Level
  - undergrad students
- Course Topics
  - Fundamentals of Compter Programing
- Course Objectives
  - Learn general concepts of programming and programming style
  - Learn how to code elegantly
  - Acquire Ability of Problem Solving and Program Design

We will use C/C++ as a programing tool

Evaluation

(1)중간시험 (Midterm Exam)	(2)기말시험 (Final Exam)	(3)출석 (Attendance)	(4)과제물 (Assignment)	(5)기타 (발표 및 토론, 프로젝트, 수업참여도 등) (Etc.(Presentation, discussion, project, participation))
30%	30%	10%	10%	20%

- Attendance below 70% will disqualify you from final exam
- Academic dishonesty (e.g. cheating, plagiarism, and etc.) will be taken seriously.

Professor Information

- Office : 424
- Email : mbilal@hufs.ac.kr
- Office Tel : 031-330-4366

Announcement

- For class material we will use eclass platform

- Programming Assignments
  - We encourage to study and discuss together for doing programming assignments.
  - However, you must do programming YOURSELF.
  - You must not share any of source code with other students.
  - Any kind of academic dishonesty will be taken seriously.

Course Objectives

Objectives of this course are three fold

1. To appreciate the need for a programming language
2. To introduce the concept and usability of the structured progra mming methodology
3. To develop proficiency in making useful software using the C++ language

Course Contents

To achieve our first two objectives we will be discussing

- Basic Programming constructs and building blocks
- Structured programming
- Structured flowcharts, pseudo-code

Course Contents

- Variables and expressions in C
- Control structures and functions
- Arrays and Pointers
- Introduction to streams
- Dynamic memory Allocation

Course Contents

- File handling
- Structures and Unions
- Flavor of Object oriented programming

Text Books

- Deitel & Deitel :– C++ How to Program
- Kernighan and Ritchie:-The C Programming Language

Goorm system

- Most of the assignments and practice in C/C++ will be done using goorm system
- TA will guide you in coming lecture

What is a program?

- Use your smart devices, you have 5 min to find the answer
- After 5 min everyone should put the device on desk

Program

“A precise sequence of steps to solve a particular problem”

Alan Perlis – Yale University:

"It goes against the grain of modern education to teach children to program. What fun is there in making plans, acquiring discipline in organizing thoughts, devoting attention to detail and learning to be self-critical? "

What are the required critical skills?

- Analysis
- Critical Thinking
- Attention to Detail

Design Recipe

To design a program properly, we must:

- **Analyze a problem statement, typically expressed as a word problem**
- **Express its essence, abstractly and with examples**
- **Formulate statements and comments in a precise language**
- **Evaluate and revise the activities in light of checks and tests**

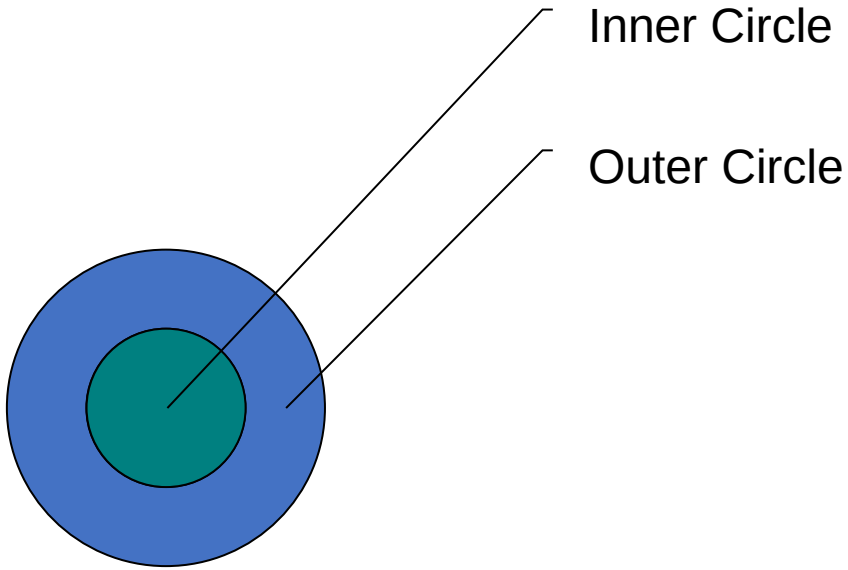
- **PAY ATTENTION TO DETAIL**
- **These skills are useful for anybody**
- **All assignments in this course should follow the these guidelines**

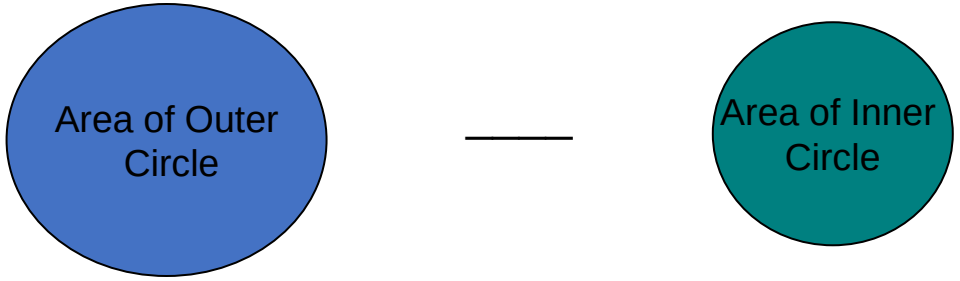
Computers are STUPID

Humans are even more…….

Think Reuse

Area of the Ring



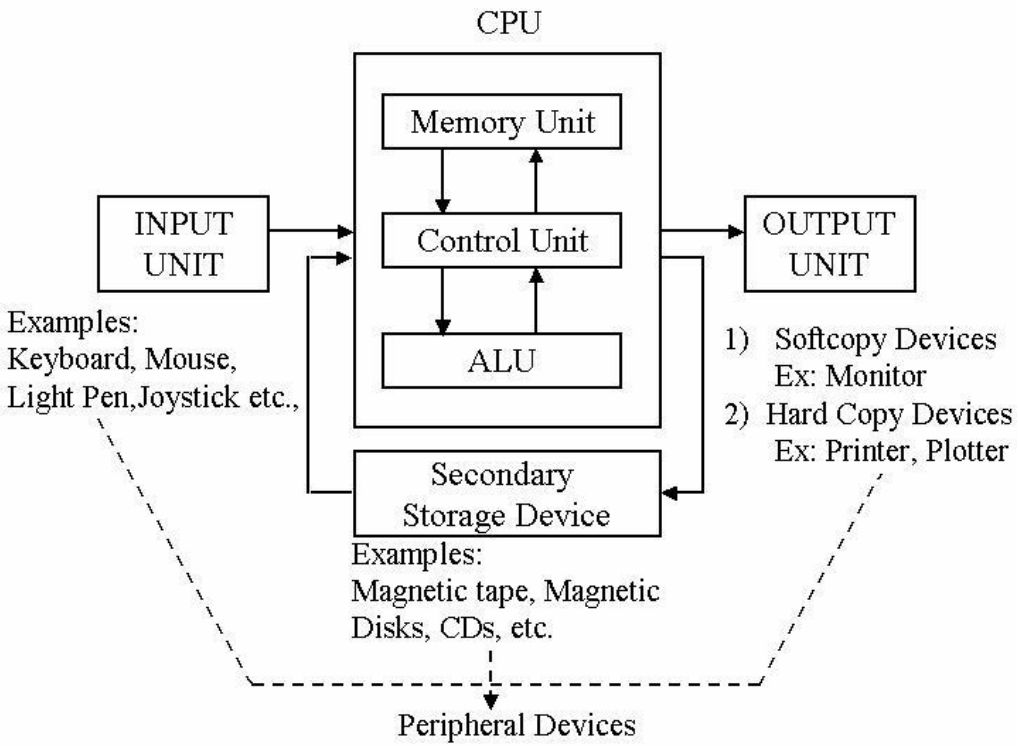


= Area of the Ring

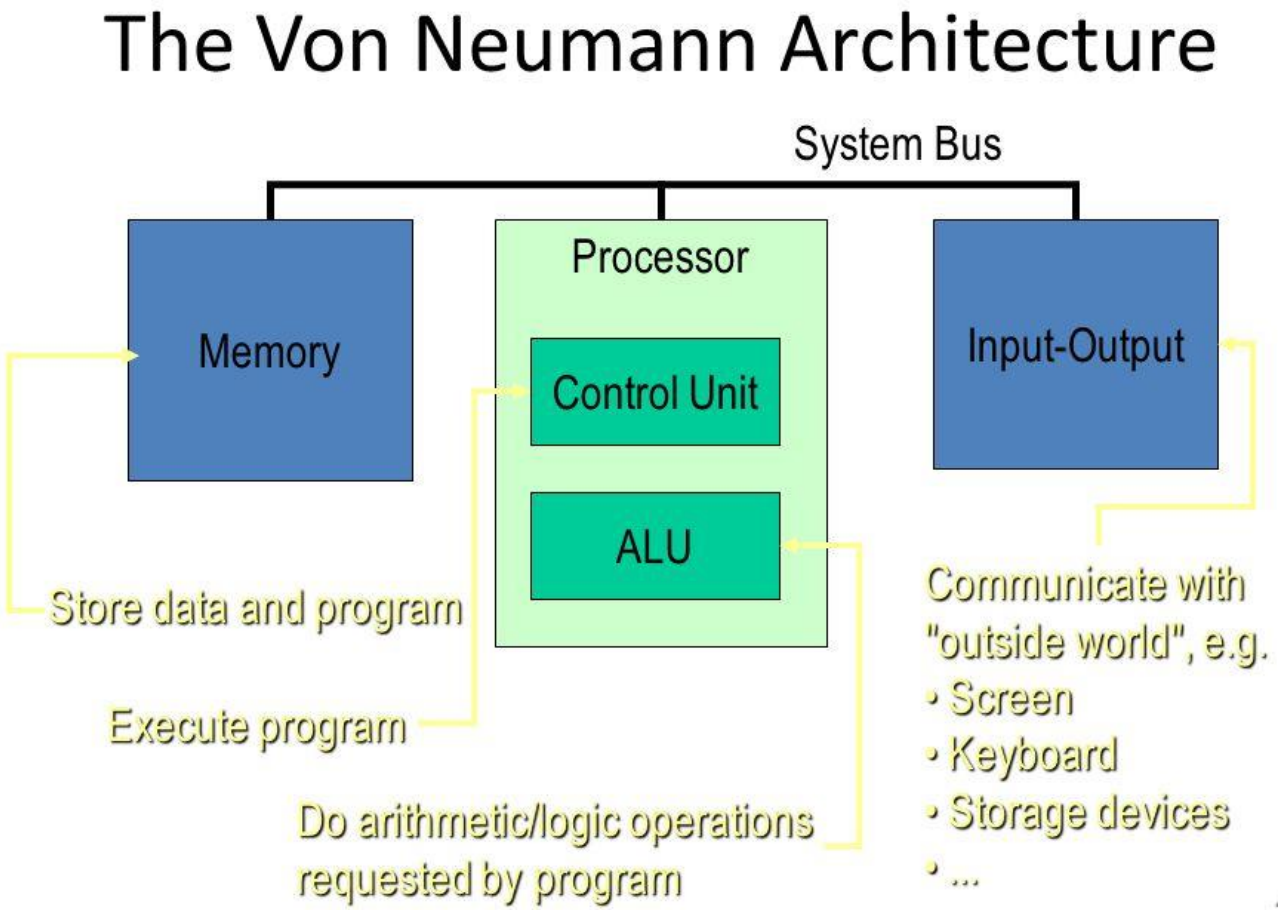
- **Think Reuse**
- **Think User Interface**
- **Comments liberally**

Computer System

- Computer System
  - Hardware + System software + Application software
- Computer Hardware

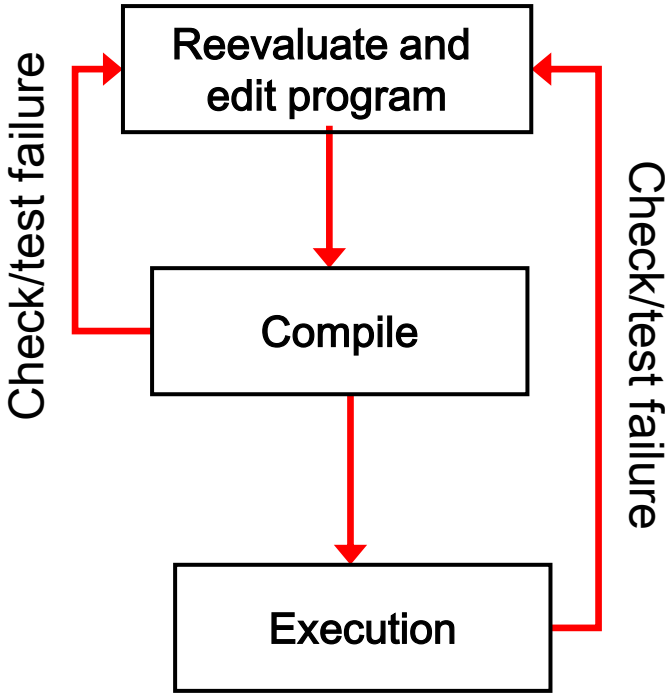


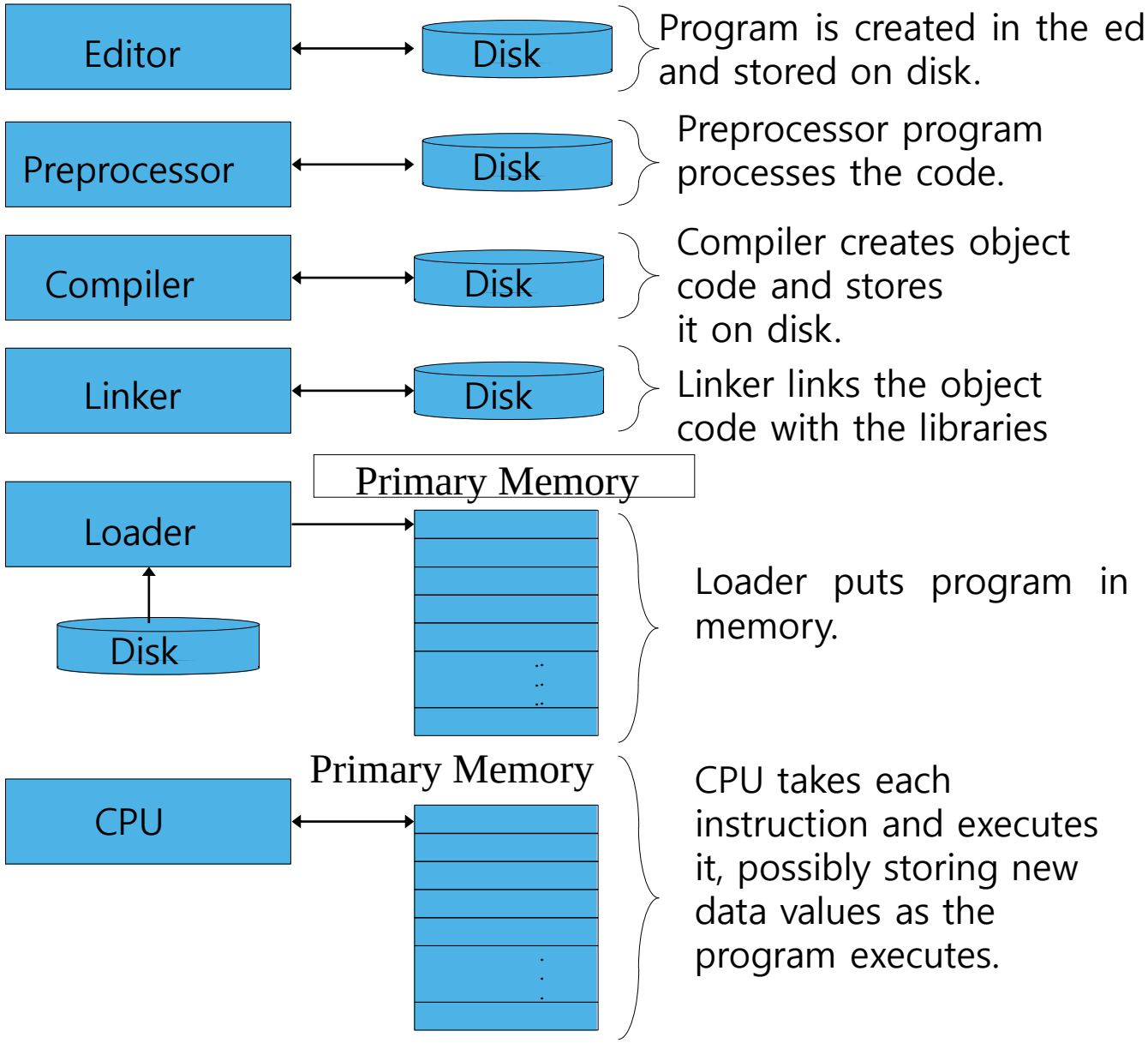
Program Execution



Software Development Process

- Requirement analysis
- Design
- Implementation
- Testing
- Maintenance




Editor
Disk
Program is created in the editor and stored on disk.
Preprocessor
Disk
Preprocessor program processes the code.
Compiler
Disk
Compiler creates object code and stores it on disk.
Linker
Disk
Linker links the object code with the libraries
Primary Memory
Loader
Disk
Loader puts program in memory.
Primary Memory
CPU
CPU takes each instruction and executes it, possibly storing new data values as the program executes.

Error

- **compile-time error**
  - Error occurring during compilation
  - Syntax check
  - Cannot execute if there is compile error

- **logical error**
  - No syntactical error but statement doesn't make sense

- **run-time error**
  - Abnormal termination owing to unexpected reasons during program execution
  - Ex) divided by zero, illegal memory access

Lewis Carol: "Through the Looking Glass"

"Twas brillig, and the s lithy toves
Did gyre and gimble i n the wabe "

Compiler vs Interpreter

- Compiler
  - Convert high level language to low level language at compile-time
- Interpreter
  - Compile and execute the program line by line at run-time

Computer Languages

- Machine language
  - Only language computer directly understands
  - Defined by hardware design
    - Machine-dependent
  - Generally consist of strings of numbers
    - Ultimately 0s and 1s
  - Instruct computers to perform elementary operations
    - One at a time
  - Cumbersome for humans
  - Example:

```
+1300042774
+1400593419
+1200274027
```

Computer Languages

- Assembly language
  - English-like abbreviations representing elementary computer operations
  - Clearer to humans
  - Incomprehensible to computers
    - Translator programs (assemblers)
      - Convert to machine language
  - Example:

```
LOAD    BASEPAY
ADD     OVERPAY
STORE   GROSSPAY
```

Computer Languages

- High-level languages
  - Similar to everyday English, use common mathematical notations
  - Single statements accomplish substantial tasks
    - Assembly language requires many instructions to accomplish simple tasks
  - Translator programs (compilers)
    - Convert to machine language
  - Interpreter programs
    - Directly execute high-level language programs
  - Example:

```
grossPay = basePay + overTimePay
```

History of C and C++

- History of C
  - Evolved from two other programming languages
    - BCPL and B
      - “Typeless” languages
  - Dennis Ritchie (Bell Laboratories)
    - Added data typing, other features
  - Development language of UNIX
  - Hardware independent
    - Portable programs
  - 1989: ANSI standard
  - 1990: ANSI and ISO standard published
    - ANSI/ISO 9899: 1990

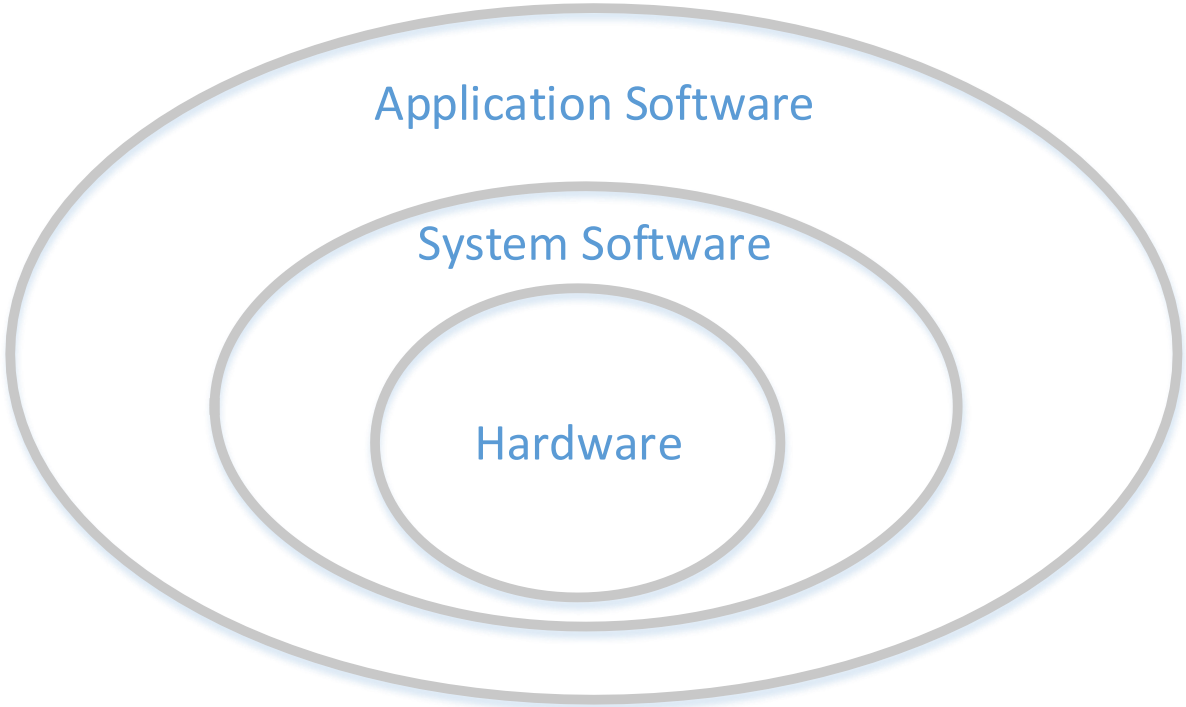
History of C and C++

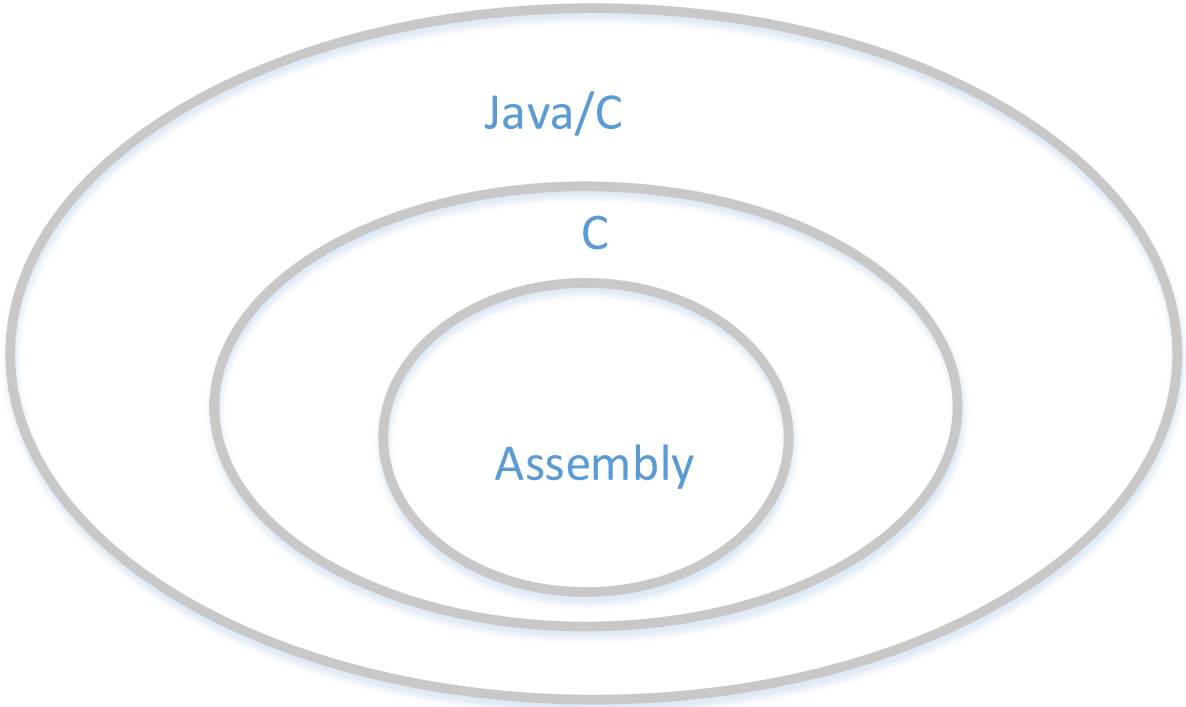
- History of C++
  - Extension of C
  - Early 1980s: Bjarne Stroustrup (Bell Laboratories)
  - Provides capabilities for object-oriented programming
    - Objects: reusable software components
      - Model items in real world
    - Object-oriented programs
      - Easy to understand, correct and modify
  - Hybrid language
    - C-like style
    - Object-oriented style
    - Both

C++ Standard Library

- C++ programs
  - Built from pieces called classes and functions
- C++ standard library
  - Rich collections of existing classes and functions
- “Building block approach” to creating programs
  - “Software reuse”

Power of C language





Power comes with responsibility

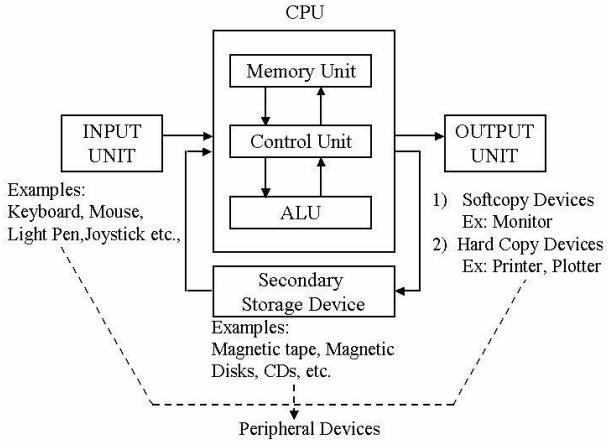
Basics of a Typical C/C++ Environment

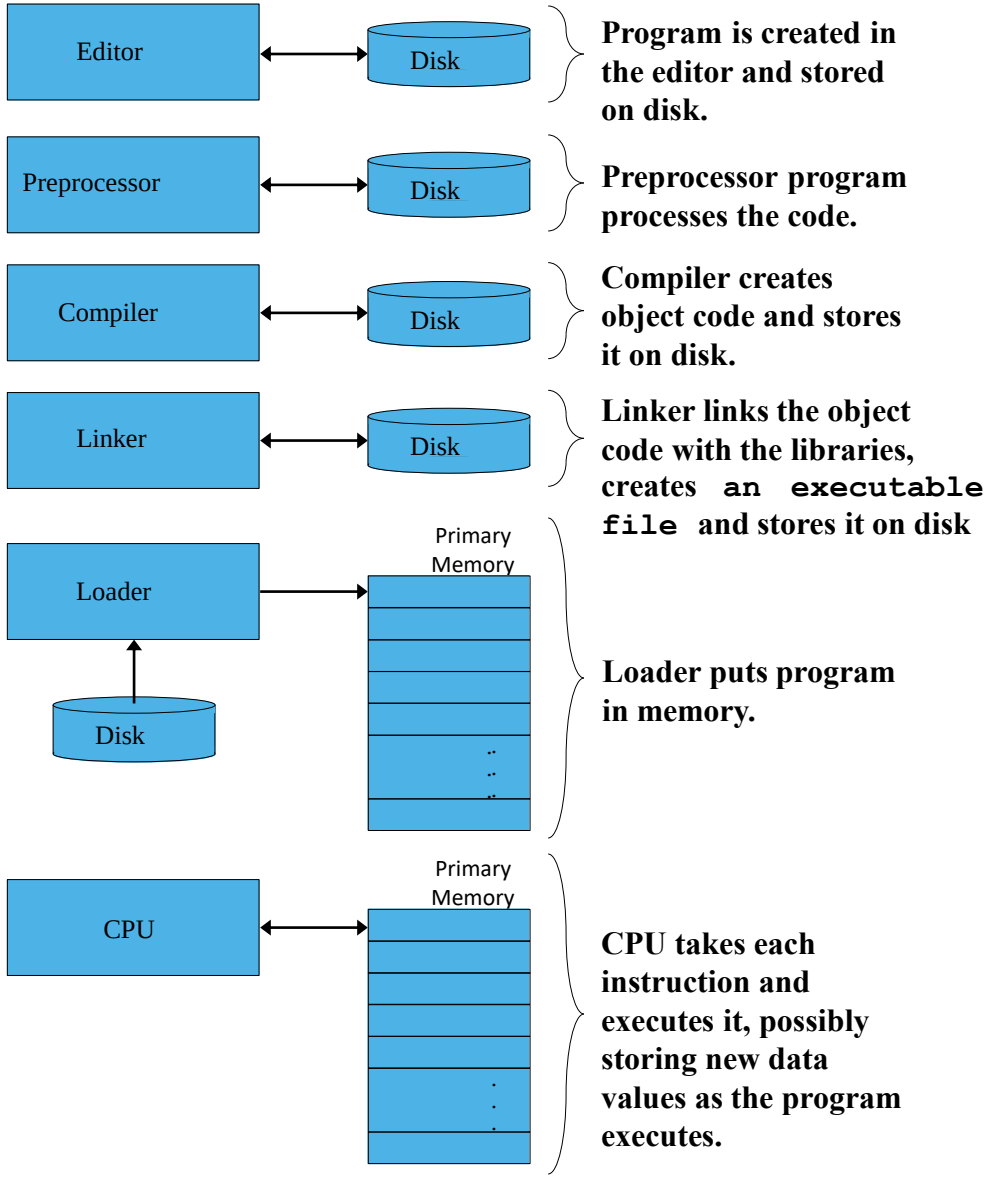
- C systems
  - Program-development environment
  - Language
  - C/C++ Standard Library
- C/C++ program names extensions
  - .cpp
  - .cxx
  - .cc
  - .C
  - .c

Basics of a Typical C/C++ Environment

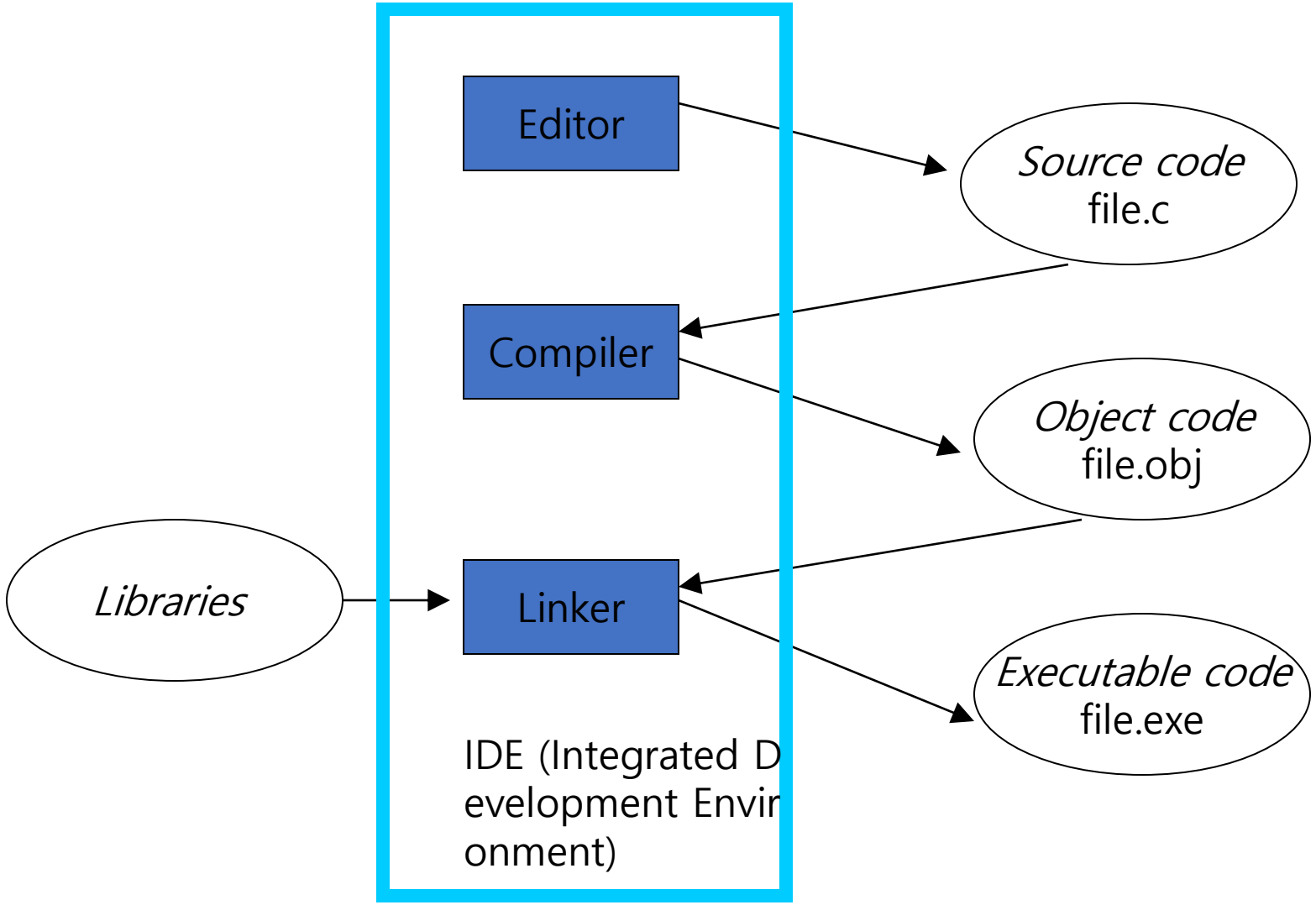
Phases of C Programs:

1. Edit
2. Preprocess
3. Compile
4. Link
5. Load
6. Execute

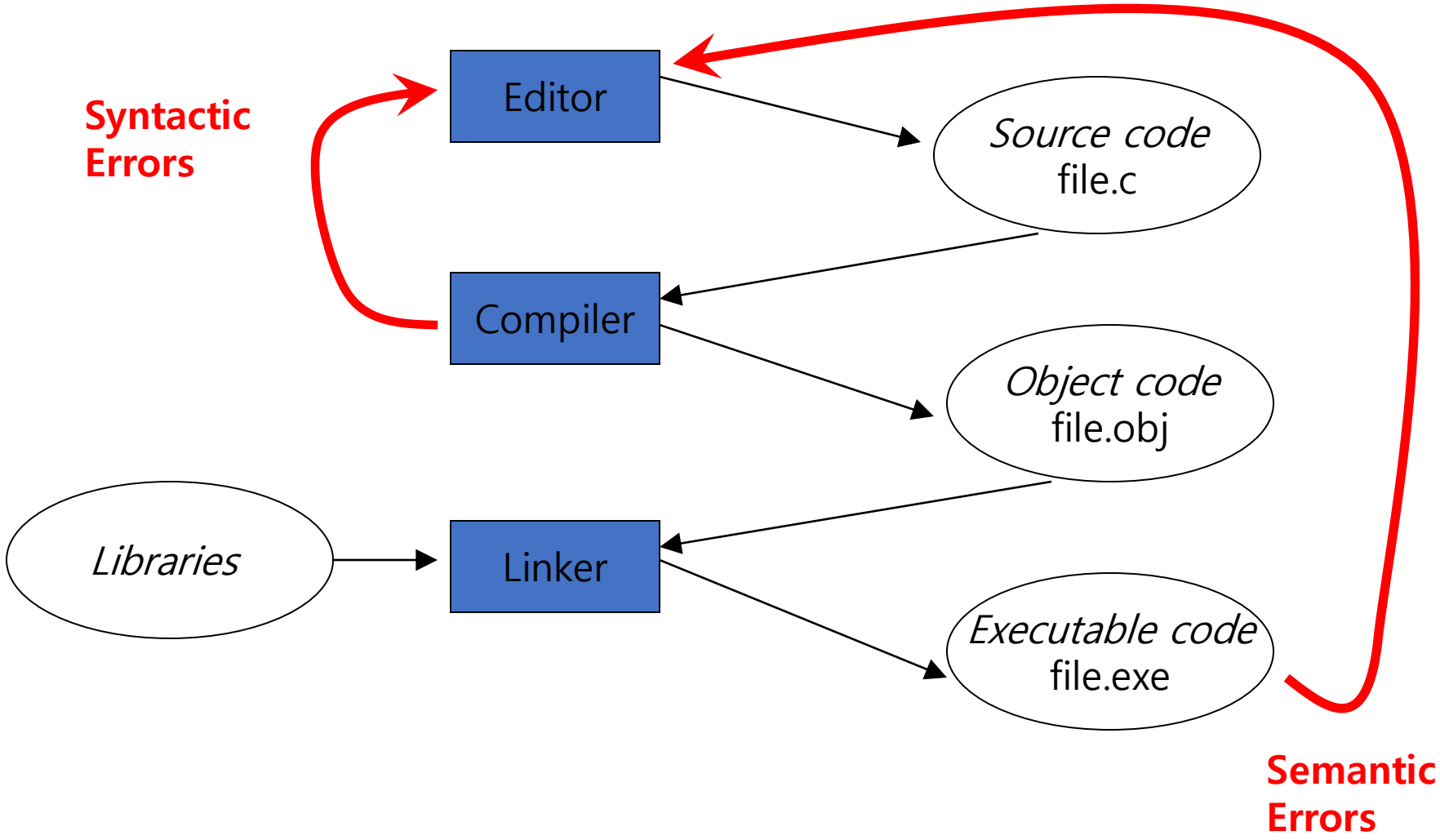




Compiling and running C++ programs



Debugging program errors



A Simple Program: Printing a Line of Text

- Before writing the programs
  - Comments
    - Document programs
    - Improve program readability
    - Ignored by compiler
    - Single-line comment
      - Use C's comment /* .. */ OR Begin with **`// or`**
  - Preprocessor directives
    - Processed by preprocessor before compiling
    - Begin with **#**

The first C++ program

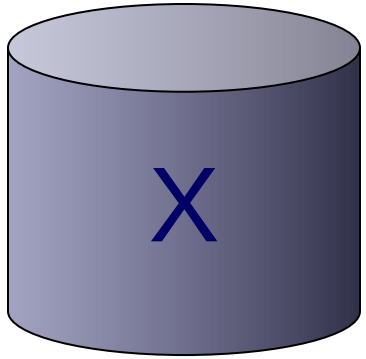
```
#include <iostream.h>
main ( )
{
  cout << " Welcome to HUFS";
}
```

A Simple Program: Printing a Line of Text

Escape Sequence	Description
₩n	Newline. Position the screen cursor to the beginning of the next line.
₩t	Horizontal tab. Move the screen cursor to the next tab stop.
₩r	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
₩a	Alert. Sound the system bell.
₩₩	Backslash. Used to print a backslash character.
₩"	Double quote. Used to print a double quote character.

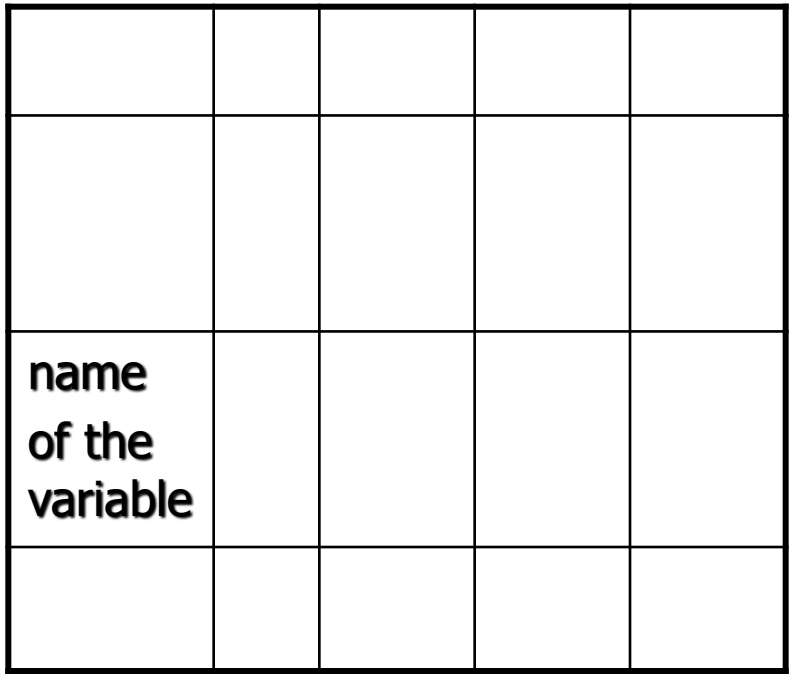
Variable

Variable



Variable

- Pic of the memory
- 25
- 10323



Variable

Variable starts with

1. Character
2. Underscore _ **(Not Recommended)**

Variable

In a program a variable has:

1. **Name**
2. **Type**
3. **Size**
4. **Value**

Variable

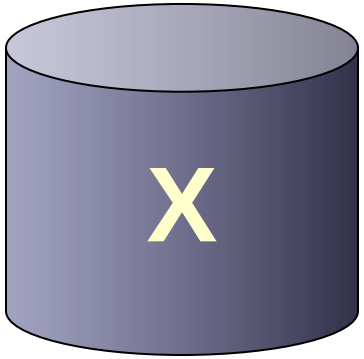
Variable is the name of a location in the memory

e.g. x= 2;

Assignment Operator

=

x = 2



2

Assignment Operator

L.H.S = R.H.S.

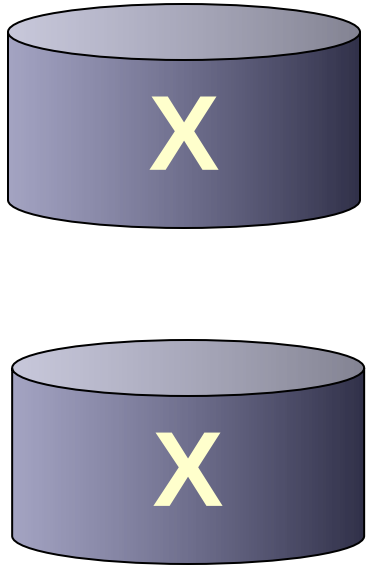
X+ 3 = y + 4 **Wrong**

Z = x +4

x +4 = Z **Wrong**

X = 10 ;

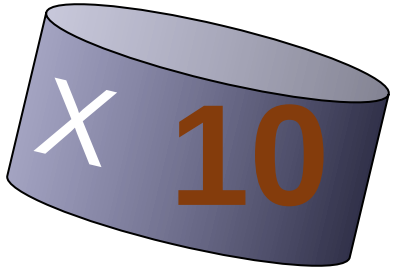
X = 30 ;



10

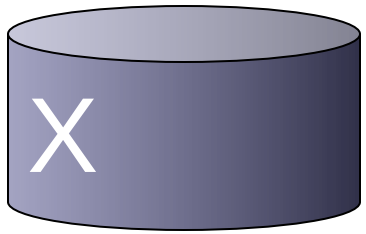
30

X = X + 1;



+ 1

11

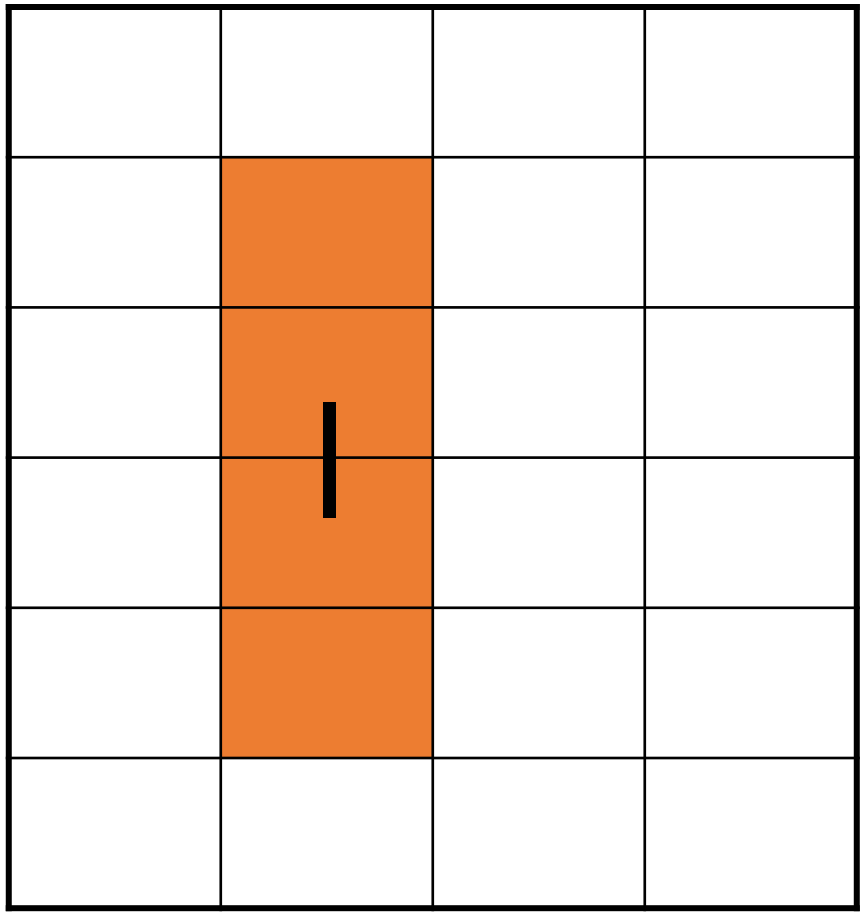


Data Types

1. **int**
2. **short**
3. **long**
4. **float**
5. **double**
6. **char**

Data type

- **int i ; -> Declarati on line**



```
/* This program adds two integer values
and displays the results */

#include <stdio.h>
#include <iostream.h>
int main (void)
{
    // Declare variables
    int value1, value2, sum;
    // Assign values and calculate their sum
    value1 = 50;
    value2 = 25;
    sum = value1 + value2;
    // Display the result
    cout<< "The sum of <<value1<<" and "<<value2<<" is "<<sum<<endl;
    printf ("The sum of %i and %i is %i\n", value1, value2, sum);
    return 0;
}
```

function vs streams

Basic Data Types - Summary

Type	Meaning	Constants Ex.	printf
int	Integer value; guaranteed to contain **at least 16 bits**	`12, -7, 0xFFE0, 0177`	`%i,%d, %x, %o`
short int	Integer value of reduced precision; guaranteed to contain **at least 16 bits**	-	`%hi, %hx, %ho`
long int	Integer value of extended precision; guaranteed to contain **at least 32 bits**	`12L, 23l, 0xffffL`	`%li, %lx, %lo`
long long int	Integer value of extraextended precision; guaranteed to contain **at least 64 bits**	`12LL, 23ll, 0xffffLL`	`%lli, %llx, %llo`
unsigned int	Positive integer value; can store positive values up to twice as large as an int; guaranteed to contain at least 16 bits (all bits represent the value, no sign bit)	`12u, 0XFFu`	`%u, %x, %o`
unsigned short int		-	`%hu, %hx, %ho`
unsigned long int		`12UL, 100ul, 0xffeeUL`	`%lu, %lx, %lo`
unsigned long long int		`12ull, 0xffeeULL`	`%llu, %llx, %llo`

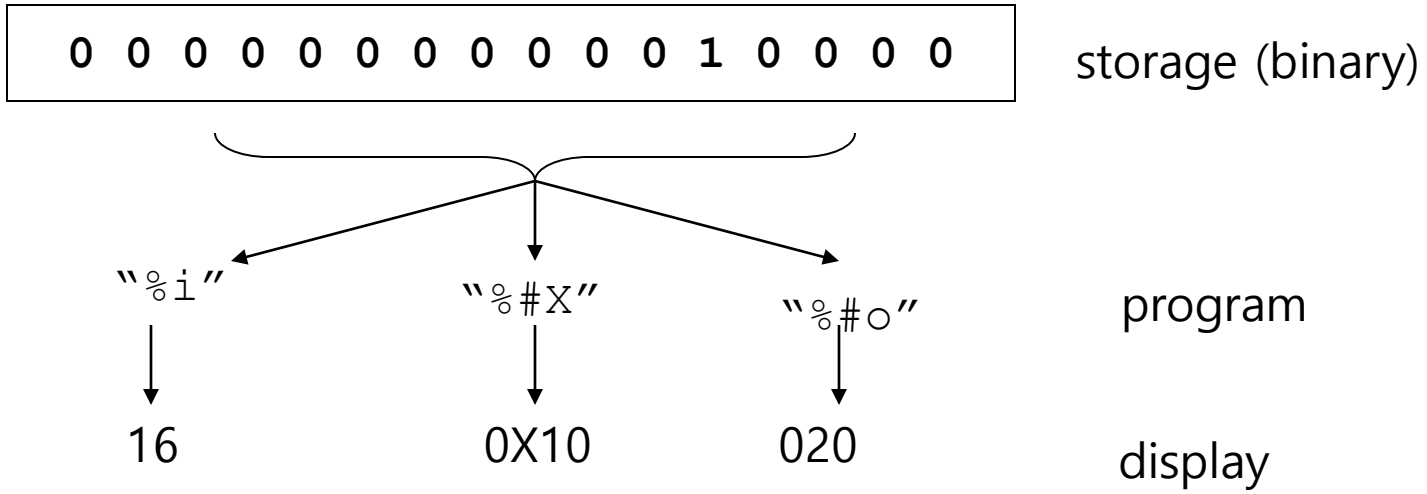
Basic Data Types - Summary (contd.)

Type	Meaning	Constants	printf
float	Floating-point value; a value that can contain decimal places; guaranteed to contain at **least six digits of precision**.	`12.34f, 3.1e-5f`	`%f, %e, %g`
double	Extended accuracy floating-point value; guaranteed to contain **at least 10 digits of precision**.	`12.34, 3.1e-5,`	`%f, %e, %g`
long double	Extraextended accuracy floating-point value; guaranteed to contain **at least 10 digits of precision.**	`12.341, 3.1e-5l`	`%Lf, %Le, %Lg`
char	Single character value; on some systems, sign extension might occur when used in an expression.	`'a', '\n'`	`%c`
unsigned char	Same as char, except ensures that sign extension does not occur as a result of integral promotion.	-	
signed char	Same as char, except ensures that sign extension does occur as a result of integral promotion.	-	

Data display vs data storage

- **We can display data in decimal, octal or hexadecimal notation, but it doesn't aff ect how the number is actually stored internally !**
- When/where to use octal and hexa: to express computer-related values in a more conv enient way

```
int x =16;
printf("%i %#X %#o\n", x,x,x);
```



Variable declarations

Data type	Variable name

User def
ined?

Which data types
are possible in C++ ?

Which variable names
are allowed in C++ ?

```
/* This program take two numbers as input,adds values
and displays the results */
#include <iostream.h>
main ( )
{
  int x ;
  int y ;
  int z ;
  x = 10 ;
  y = 20 ;
  z = x + y ;
  cout << " x = " ;
  cout << x ;
  cout << " y = " ;
  cout << y ;
  cout << " z =x + y = " ;
  cout << z ;
}
```