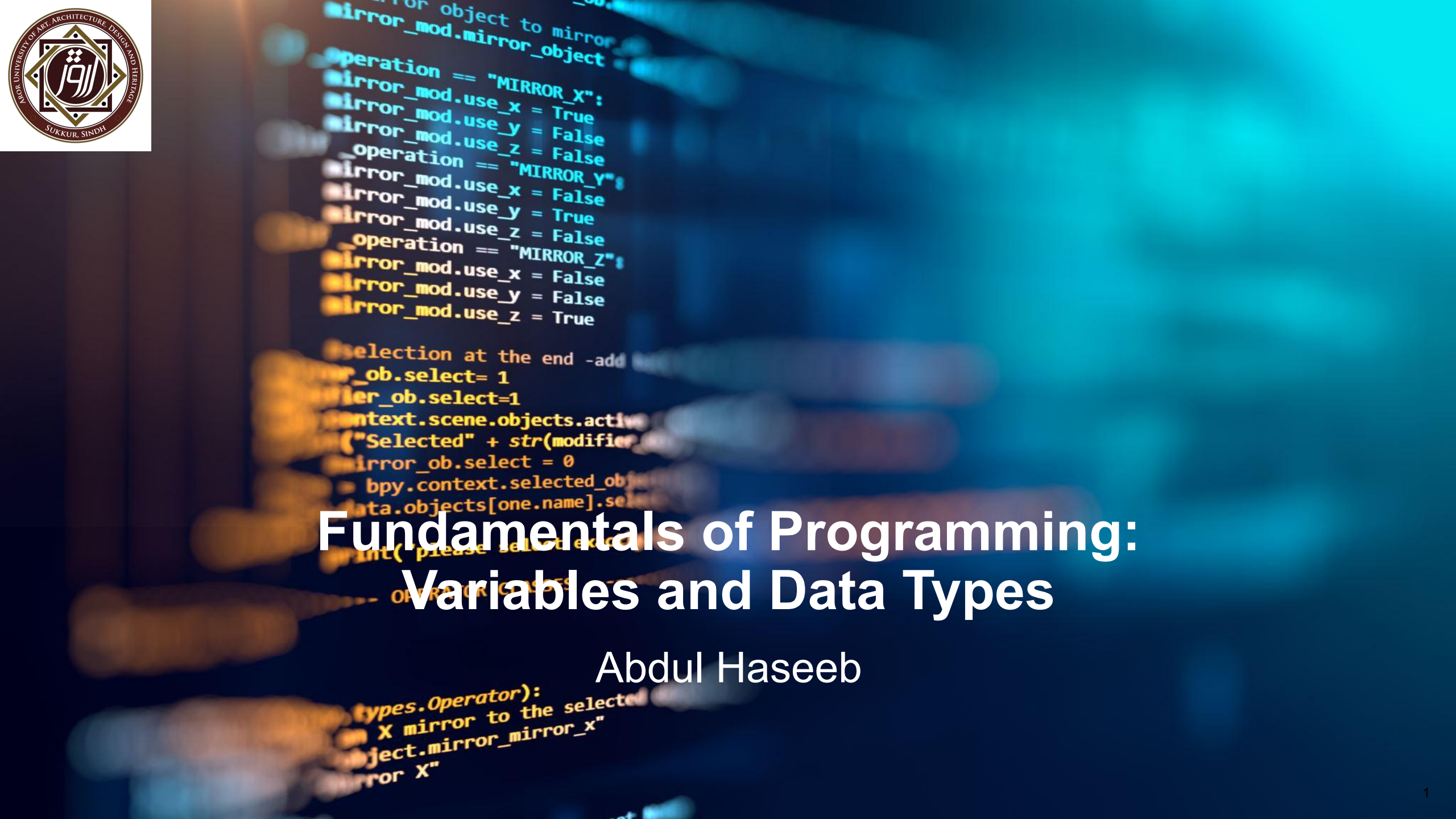


Fundamentals of Programming: Variables and Data Types

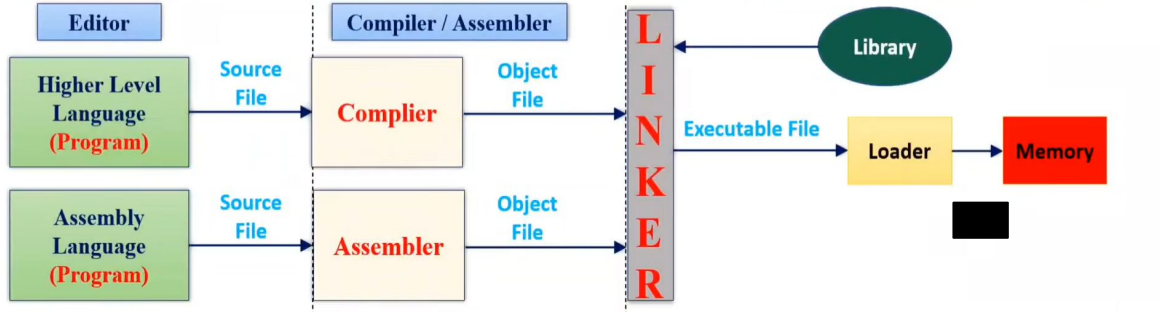
Abdul Haseeb

RECAP

- IDE
- Problem Solving
- Algorithms
- Structure of first C++ Program
- Features of C++

Processing of a C++ Program

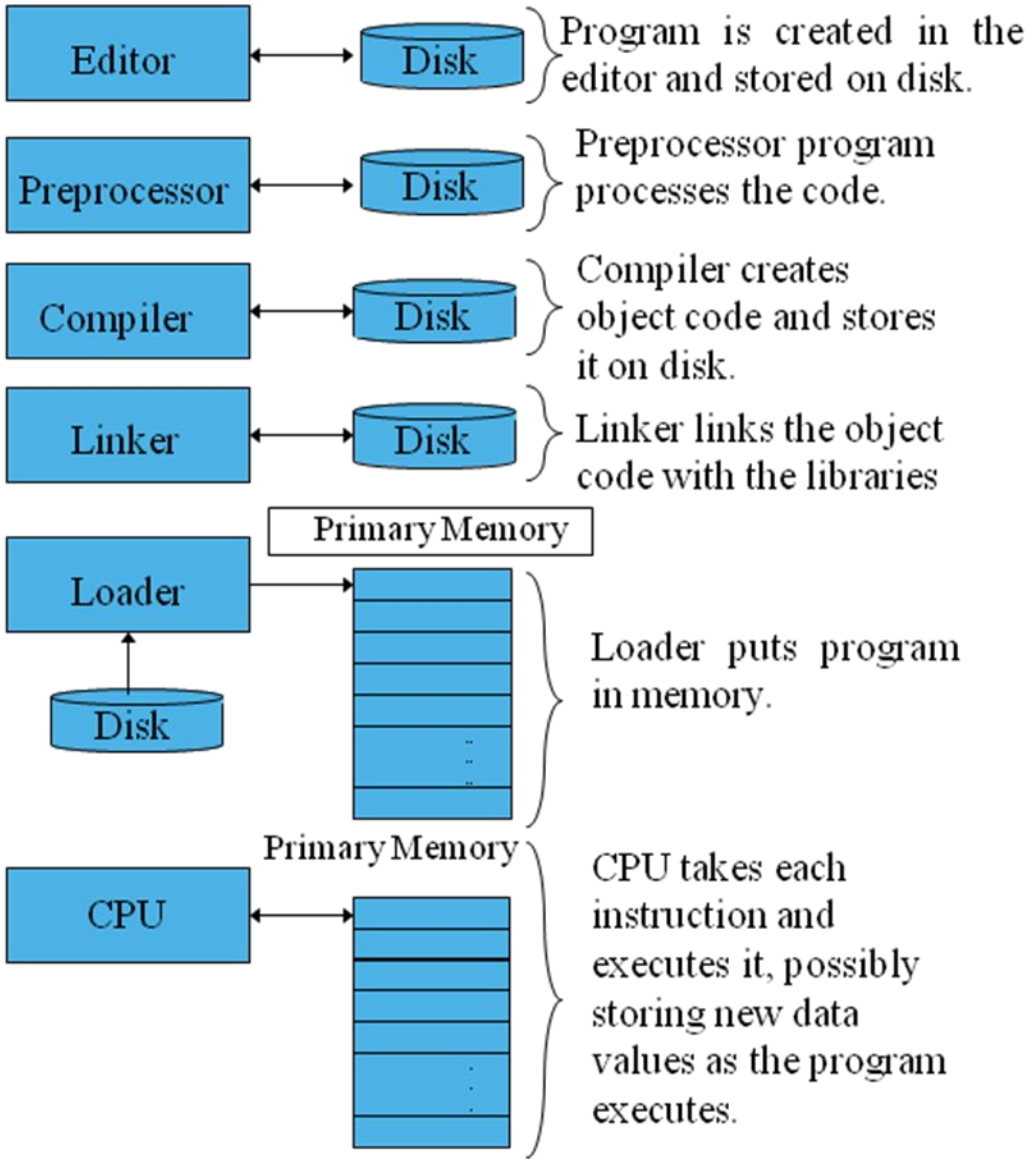
Editor, Compiler, Assembler, Linker & Loader



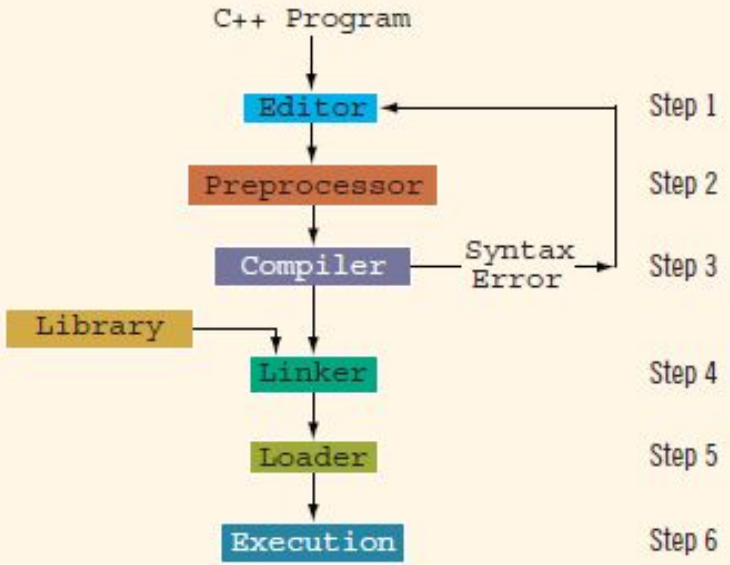
- In Editor we write programs for Microcontroller.
- Programs may be written in Assembly Language or Higher Level Language {C Language}.
- By writing program we generate source file.
- Assembler : It is used to convert Assembly language into machine code or object file. It also shows errors if any syntax error is there in program.
- Complier : It is used to convert Higher Level language into machine code or object file. It also shows errors if any syntax error is there in program. It also gives warnings if it is there with programs.

- Linker : It is linking all the object files of compiler and assembler with the use of library.
- It will generate executable files.
- Loader: It is used to load executable files into the memory of microcontroller.
- Once program is loaded into memory, microcontroller can execute it as per the requirement of USER.

Processing of a C++ Program



Processing of a C++ Program



Variables

- We often use **different types of data**
  - Variables are used:
    - To store value for a particular type of data

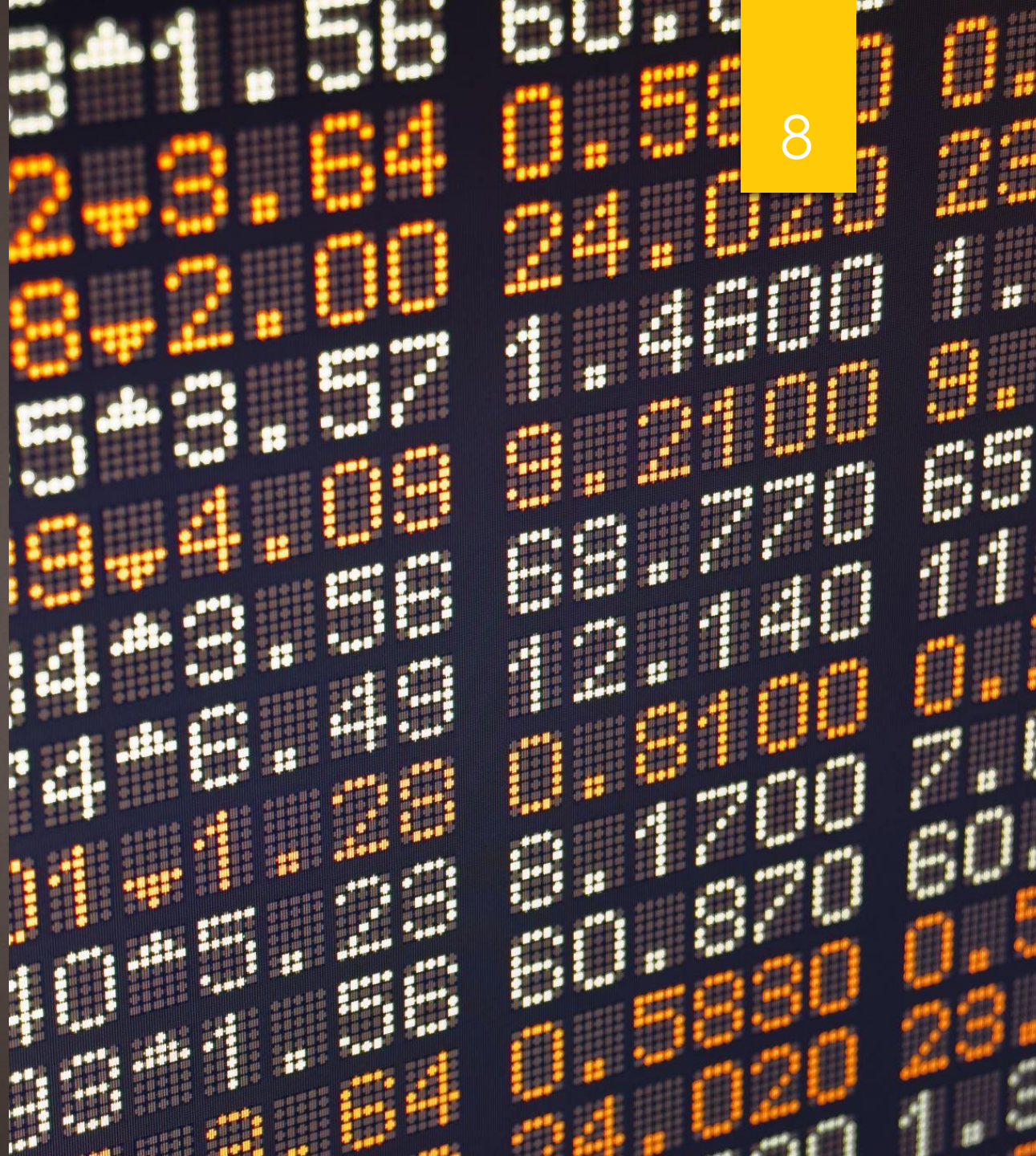
Variables

- Each variable in C++ has:
  - Data Type
  - Name
  - Value

Three Actions for Variables

- Declaring a Variable:
  1. Set the name and data type for a variable
     a) These two properties don't change

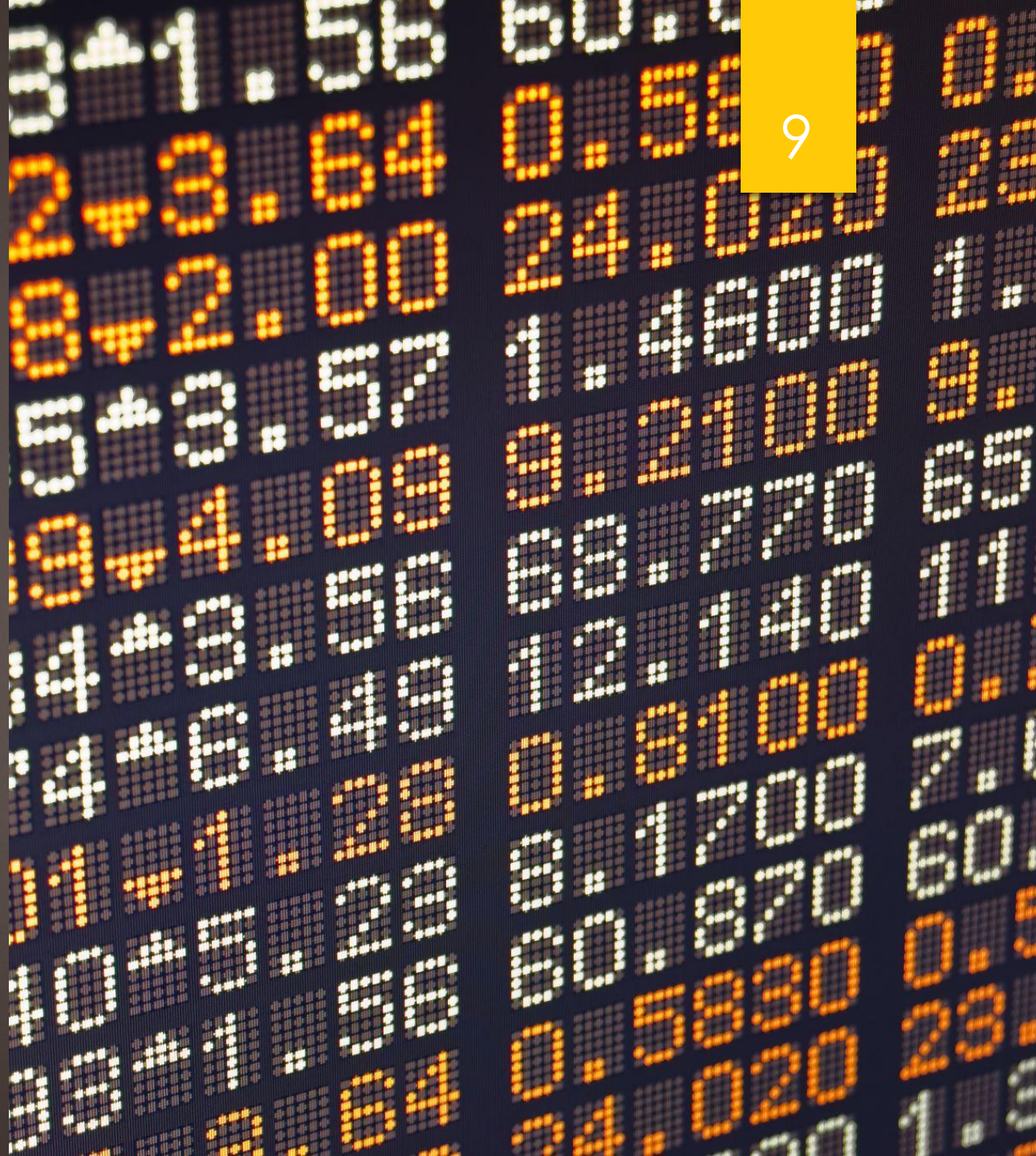
```
//declaring (giving the variable a type)
int number;
bool true_or_false;
char letter;
```



Three Actions for Variables

- Assigning (Initializing) a Variable:
  1. Set the value of a variable
     a) It can change

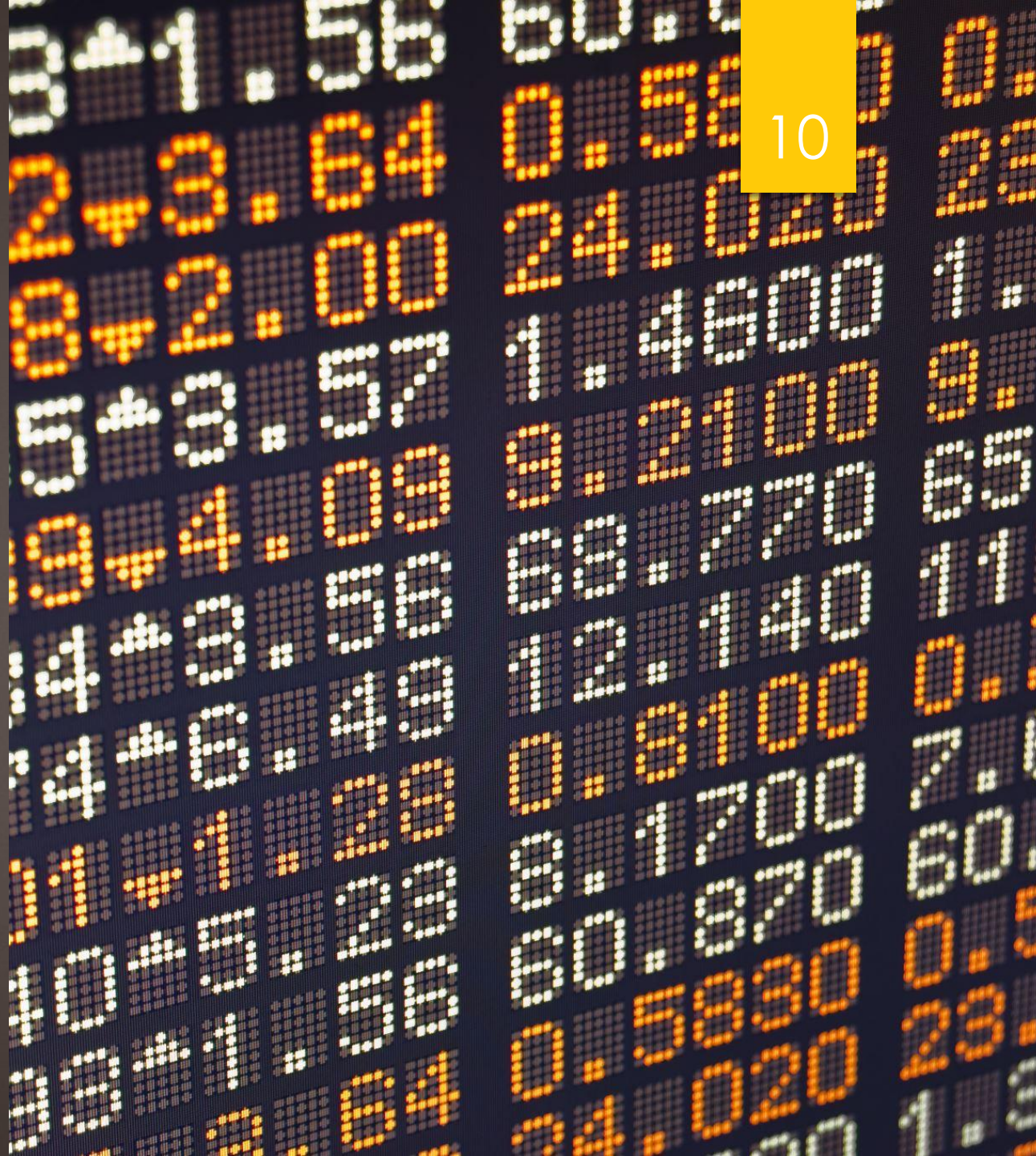
```
//assigning (giving the variable a value)
number = 99;
true_or_false = true;
letter = 'a';
```



Three Actions for Variables

- Accessing a Variable:
  1. Retrieve the value, by calling it's name.
  2. You must declare and assign a variable before you can access it.

```
//accessing (retrieving the value of the data by printing)
cout <<number <<endl;
cout << true_or_false << endl;
cout << letter << endl;
```



Characteristics of a Variable

- Variables are changeable
- Variables are container
- Variables are identifier
- Example: a, name, age, salary, x

Variable and its values in memory

- Variable: a memory location whose contents can be changed

Figure 2-2 Memory allocation

Rules for defining a variable name

A to Z

a to z

Alphanumerical like a1, day1, name1 etc

Multiple characters like name, fName, etc.

Can not use special characters except '_' underscore.

Can not be keywords of C/C++

Examples

- a Correct
- A Correct
- Age Correct
- name Correct
- name1 Correct
- first_name Correct
- _age Correct
- first-name Incorrect
- 2age Incorrect
- my@age Incorrect
- include Incorrect
- Include Correct
- delay Incorrect

Reserved words (Keywords) in C++

- Reserved word symbols (or keywords):
  - Cannot be redefined within program
  - Cannot be used for anything other than their intended use

  Examples:
    - `int`
    - `float`
    - `double`
    - `char`
    - `const`
    - `void`
    - `return`

What Are C++ Key Words?

C++ keys words are words that are reserved for specific functions or tasks within C++ programs. These words **cannot** be used to name variables and will result in errors if they are not handled correctly. Click below to see a list of C++ key words.

▼ **List of C++ key words**

and	and_eq	asm	auto	bitand
bitor	bool	break	case	catch
char	class	compl	const	const_cast
continue	default	delete	do	double
dynamic_cast	else	enum	explicit	extern
false	float	for	friend	goto
if	inline	int	long	mutable
namespace	new	not	not_eq	operator
or	or_eq	private	protected	pubic
register	reinterpret_cast	return	short	signed
sizeof	static	static_cast	struct	switch
template	this	throw	true	try
typedef	typeid	typename	union	unsigned
using	virtual	void	volatile	wchar_t
while	xor	xor_eq		

Constants

- Constants are those variables whose values can not be changed
- Fix in nature
- Example:
  - 9.8
  - 3.14

Using const and #define for making constants

Literals in C++

- Literals are values that are assigned to variables or constants
  - Integer literals
  - Floating point literals
  - Character literals
  - String literals
  - Boolean literals

Whitespaces

- Every C++ program contains whitespaces
  - Include blanks, tabs, and newline characters
- Used to separate special symbols, reserved words, and identifiers
- Proper utilization of whitespaces is important
  - Can be used to make the program more readable

Data Types

- Shows the type of Data
- Example:
- 92 (Numeric data)
- 9.8 (Decimal Data)
- A (Character Data)
- Mujtaba (String / Wording Data)

Why different data types?

Data Representation

Memory efficiency

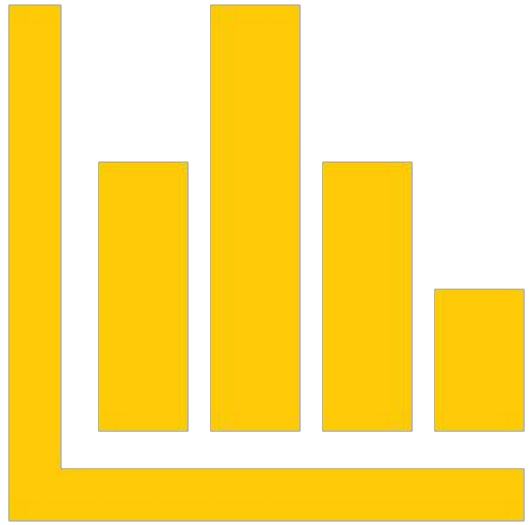
Different operations and functions (arithmetic on Integers, Comparison on Text Values)

Data Types

- C++ data types fall into three categories:
  - Simple data type
  - Structured data type
  - Pointers

Simple Data Types

- There are four simple data types in C / C++.
  - Countable data types
  - Measurable data types
  - Character data types
  - String data types



Signed vs Unsigned Number

- Unsigned numbers include 0 or positive numbers
- Signed Numbers include 0, positive and negative numbers

Countable Data Types

- Integer: Whole Numbers, Can be positive or negative
  - unsigned integer
  - signed integer
  - unsigned short
  - signed short
  - unsigned long
  - signed long

Countable Data Types


```cpp
#include <iostream>
using namespace std ;
int main ()
{
    short s1 = 10 ;
    signed short s2 = 20 ;
    unsigned short s3 = 30 ;
    int i1 = 10 ;
    signed int i2 = 20 ;
    unsigned int i3 = 30 ;
    long l1 = 10 ;
    signed long l2 = 20 ;
    unsigned long l3 = 30 ;
    long long ll1 = 10 ;
    signed long long ll2 = 20 ;
    unsigned long long ll3 = 30 ;
    cout <<"The s1 size in memory = " <<sizeof (s1) <<endl;
    cout <<"The s2 size in memory = " <<sizeof (s2) <<endl;
    cout <<"The s3 size in memory = " <<sizeof (s3) <<endl;
    cout <<"The i1 size in memory = " <<sizeof (i1) <<endl;
    cout <<"The i2 size in memory = " <<sizeof (i2) <<endl;
    cout <<"The i3 size in memory = " <<sizeof (i3) <<endl;
    cout <<"The l1 size in memory = " <<sizeof (l1) <<endl;
    cout <<"The l2 size in memory = " <<sizeof (l2) <<endl;
    cout <<"The l3 size in memory = " <<sizeof (l3) <<endl;
    cout <<"The ll1 size in memory = " <<sizeof (ll1) <<endl;
    cout <<"The ll2 size in memory = " <<sizeof (ll2) <<endl;
    cout <<"The ll3 size in memory = " <<sizeof (ll3) <<endl;
    return 0 ;
}
```

```
The s1 size in memory = 2
The s2 size in memory = 2
The s3 size in memory = 2
The i1 size in memory = 4
The i2 size in memory = 4
The i3 size in memory = 4
The l1 size in memory = 4
The l2 size in memory = 4
The l3 size in memory = 4
The ll1 size in memory = 8
The ll2 size in memory = 8
The ll3 size in memory = 8
```

Countable Data Types


```cpp
#include <iostream>
#include <climits>
using namespace std ;
int main ()
{
    cout << "\n\n Check the upper and lower limits of integer :\n";
    cout << "-------------------------------------------------\n";
    cout << " The maximum limit of int data type :                  " << INT_MAX << endl;
    cout << " The minimum limit of int data type :                  " << INT_MIN << endl;
    cout << " The maximum limit of unsigned int data type :         " << UINT_MAX << endl;
    cout << " The maximum limit of long long data type :            " << LLONG_MAX << endl;
    cout << " The minimum limit of long long data type :             " << LLONG_MIN << endl;
    cout << " The maximum limit of unsigned long long data type :   " << ULLONG_MAX << endl;
    cout << " The minimum limit of short data type :                " << SHRT_MIN << endl;
    cout << " The maximum limit of short data type :                " << SHRT_MAX << endl;
    cout << " The maximum limit of unsigned short data type :       " << USHRT_MAX << endl;
    cout << endl;
    return 0 ;
}

```

Countable Data Types


```
Check the upper and lower limits of integer :
-------------------------------------------------
The maximum limit of int data type :                2147483647
The minimum limit of int data type :                -2147483648
The maximum limit of unsigned int data type :       4294967295
The maximum limit of long long data type :          9223372036854775807
The minimum limit of long long data type :           -9223372036854775808
The maximum limit of unsigned long long data type :  18446744073709551615
The minimum limit of short data type :              -32768
The maximum limit of short data type :              32767
The maximum limit of unsigned short data type :     65535


--------------------------------
Process exited after 0.08577 seconds with return value 0
Press any key to continue . . .
```

Important Take Away

What happens if you:

- Change the variable to `5000`?
- Change the variable to `5,000`?
- Change the variable to `050`?
- Change the variable to `"5000"` (with double quotes)?

Measurable Data Types

- Floating Point Numbers: Numbers with a Decimal, can be positive or negative
  - float
  - double
  - long double

Measurable Data Types


```cpp
#include <iostream>
#include <cfloat>
using namespace std ;
int main ()
{
    float f = 9.9 ;
    double d = 9.9 ;
    long double ld = 9.9 ;
    cout << "\n\n Check the upper and lower limits of countable datatypes :\n";
    cout << " The minimum limit of float data type :                " << FLT_MIN << endl;
    cout << " The maximum limit of float data type :                " << FLT_MAX << endl;
    cout << " The minimum limit of double data type :                " << DBL_MIN << endl;
    cout << " The maximum limit of double data type :                " << DBL_MAX << endl;
    cout << " The minimum limit of long double data type :                    " << LDBL_MIN << endl;
    cout << " The maximum limit of long double data type :                    " << LDBL_MAX << endl;
    cout << endl;
    cout << "\n\n Check the space of memory occupied by countable datatypes :\n";
    cout <<"The f size in memory = " <<sizeof (f) <<endl;
    cout <<"The d size in memory = " <<sizeof (d) <<endl;
    cout <<"The ld size in memory = " <<sizeof (ld) <<endl;
    return 0 ;
}
```

Measurable Data Types


```
 Check the upper and lower limits of countable datatypes :
 The minimum limit of float data type :                1.17549e-038
 The maximum limit of float data type :                3.40282e+038
 The minimum limit of double data type :                2.22507e-308
 The maximum limit of double data type :                1.79769e+308
 The minimum limit of long double data type :                3.3621e-4932
 The maximum limit of long double data type :                1.18973e+4932



 Check the space of memory occupied by countable datatypes :
The f size in memory = 4
The d size in memory = 8
The ld size in memory = 16
```

What happens if you:

- Change the variable to `50.`?
- Change the variable to `.001`?

Boolean Data Type

- Variable can take any of the following two values:
  - true (1)
  - false (0)

Example


```
bool thisIsFun = true;
cout << boolalpha << thisIsFun << endl;
```

What happens if you:

- Change the variable to `false`?
- Remove the `boolalpha <<` command?
- Change the variable to `True`?
- Change the variable to `False`?
- Change the variable to `TRUE`?

Character Datatype

- Store single character
- In coding we use ***char*** for character datatype
- The value must be enclosed in single quotes (‘ ’)
- Syntax: char variable_name = ‘Value’ ;
- Example: char grade = ‘A’ ;
- Character datatype will take **1 byte** of memory
- Range of character is **-128 to 127**
- Range of unsigned character is **255**
- **CHAR_MIN**
- **CHAR_MAX**
- **UCHAR_MAX**

Character Datatype

```
#include <iostream>
using namespace std ;
int main ()
{
cout << CHAR_MIN <<endl ;
cout << CHAR_MAX <<endl ;
cout <<UCHAR_MAX <<endl ;
char sub1_grade = 'A' , sub2_grade = 'B+' ;
cout << sub1_grade <<endl <<sub2_grade <<endl ;
return 0 ;
}
```

Character Datatype

```
#include <iostream>
using namespace std ;
int main ()
{
char c = 65 ;
cout <<c <<endl ;
c = '65' ;
cout << c <<endl ;
c = 200 ;
cout << c <<endl ;
return 0 ;
}
```

String Datatype

- Store sequence of characters
- In coding we use ***char*** **[]** or **string** for string datatype
- The value must be enclosed in double quotes (“ “)
- Syntax: char variable_name [size in number] = “Value” ;
- Example: char std_name = “Ahmad” ;
- string variable_name = “value” ;
- string std_name = “Mujtaba” ;
- Char [***n***] datatype will take ***n*** **number of bytes** of memory.
- String datatype will take ***n*** **number of bytes** of memory where ***n*** represents the number of characters in string.

String Datatype

- **What will be the output of following program?**

```
#include <iostream>
using namespace std ;
int main ()
{
char name [10] = "Mujtaba" ;
string name1 = "Mujtaba" ;
cout << name << "\t" << sizeof (name) <<endl;
cout << name1 << "\t" << sizeof (name1) <<endl;
return 0 ;
```

What happens if you:

- Forget one of the " quotation marks?
- Forget both " " quotation marks?
- Use single (') quotation marks?
- Use uppercase `String` instead of lowercase `string`?

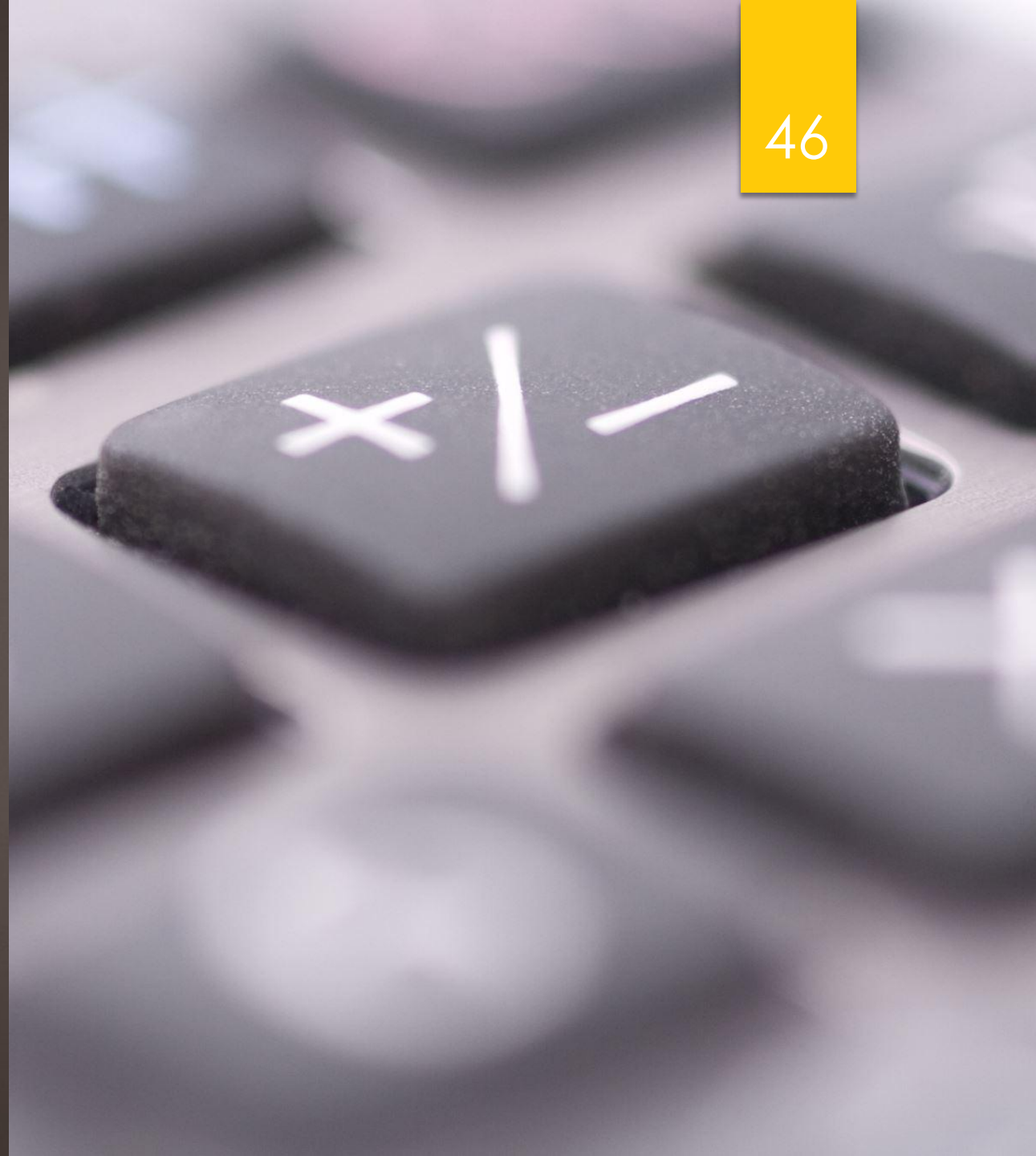
Summary of ranges and sizes for each data type

DATA TYPE	SIZE (IN BYTES)	RANGE
short int	2	-32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295
long long int	8	-(2^63) to (2^63)-1
unsigned long long int	8	0 to 18,446,744,073,709,551,615
signed char	1	-128 to 127
unsigned char	1	0 to 255
float	4	
double	8	
long double	12	
wchar_t	2 or 4	1 wide character

I/O in C++

Taking Input in C++

- **cin object:**
  - Object of iostream class just like cout
  - Accepts input from the standard device i.e keyboard
  - Extraction operator (>>) is used along with cin, for reading inputs
    - Extracts data from cin



```
// C++ program to demonstrate the
// cin object
#include <iostream>
using namespace std;

// Driver Code
int main()
{
    string s;

    // Take input using cin
    cin >> s;

    // Print output
    cout << s;

    return 0;
}
```

Demo code for cin

Demo code#02 for cin

- Taking Multiple Inputs:


```cpp
// C++ program to illustrate the take
// multiple input
#include <iostream>
using namespace std;

// Driver Code
int main()
{
    string name;
    int age;

    // Take multiple input using cin
    cin >> name >> age;

    // Print output
    cout << "Name : " << name << endl;
    cout << "Age : " << age << endl;

    return 0;
}
```

Displaying Output in C++

- **cout object:**
  - Object of iostream
  - Display output to the standard device i.e monitor
  - Uses Insertion operator **<<** :
    - Inserts the data into the standard device

More than one variable can be printed using the insertion operator(<<) with cout.

```cpp
// C++ program to illustrate printing
// of more than one statement in a
// single cout statement
#include <iostream>
using namespace std;

// Driver Code
int main()
{
    string name = "Akshay";
    int age = 18;

    // Print multiple variable on
    // screen using cout
    cout << "Name : " << name << endl
         << "Age : " << age << endl;

    return 0;
}
```

Output:

```
Name : Akshay
Age : 18
```

Manipulators in C++

- **Manipulators** are helping functions:
  - Modify the input/output stream
  - We **don't change the value of a variable**, we just **change the format of display**
  - **#include<iomanip> header file** must be included
  - Manipulators can be **without arguments** like endl
  - Manipulators can be **with arguments** like setbase(), which can take 16,8,10

Manipulators in C++

- **You can explore many more manipulators on the following link:**
  - https://www.geeksforgeeks.org/manipulators-in-c-with-examples/?ref=lbp

Escape Sequences

Definition:

- Special Non-Printing characters
- Control the printing behavior of output stream objects like cout

Syntax:

- Escape Sequences consists of two characters
  - First one is a backslah \ (escape character)
  - Second one is coded character which actually controls the printing behavior

Places where you can insert an escape sequence

Beginning of a string

In the middle of a string

At the end of a string

Some commonly used escape sequences

Escape Sequence	Name	Description
\a	Alarm or Beep	It is used to generate a bell sound in the C program.
\b	Backspace	It is used to move the cursor one place backward.
\f	Form Feed	It is used to move the cursor to the start of the next logical page.
\n	New Line	It moves the cursor to the start of the next line.
\r	Carriage Return	It moves the cursor to the start of the current line.
\t	Horizontal Tab	It inserts some whitespace to the left of the cursor and moves the cursor accordingly.

\\	Backlash	Use to insert backslash character.
\'	Single Quote	It is used to display a single quotation mark.
\"	Double Quote	It is used to display double quotation marks.
\?	Question Mark	It is used to display a question mark.

Some commonly used escape sequences

Scope of the Variables

- Scope:
  - Area of program where variable is valid and available to use
  - Mainly two types of variable scopes:
    - Local Variable
    - Global Variable

Local Variables

- Defined within a function or block
- Local variables only exist inside the function or block:
  - Can not be accessed outside that block or function

Local Variables Example

C++

```cpp
// CPP program to illustrate
// usage of local variables
#include<iostream>
using namespace std;

void func()
{
    // this variable is local to the
    // function func() and cannot be
    // accessed outside this function
    int age=18;
}

int main()
{
    cout<<"Age is: "<<age;

    return 0;
}
```

Output:

```
Error: age was not declared in this scope
```

Global variables

- Can be accessed in any part of the program
- Declared at the top of the program outside all functions and blocks

```cpp
#include<iostream>
using namespace std;

// global variable
int global = 5;

// global variable accessed from
// within a function
void display()
{
    cout<<global<<endl;
}

// main function
int main()
{
    display();

    // changing value of global
    // variable from main function
    global = 10;
    display();
}
```

Output:

```
5
10
```

What will happen if there exists a local variable with the same name as the global variable

- Normally if there are two local variables with same name then compiler generates an error
- But if there is local and global variable with the same name then local variable is given preference
  - Then how to access the global value?

Scope resolution operator

- :: is called scope resolution operator
- It allows us to access the global variable inside a function or block
- Mostly used when there is a local and global variable with the same name

Type Casting and Type conversion

- After Operators