Programming in C++

Characteristics:

1) Case sensitive language:
   e.g a, A are different

2) Alphabets:
   - a-z
   - A-Z
   - 0-9
   - Spesial symbols

3) Reserved/Key words

4) Ignores all white spaces:
   e.g

   a   +   b=   50;

   is same as

   a +b =50;

5) Comments:
   These are ~~the~~ text that are ignored by compile

   // for single line comment.

   /* ---
   - - - multiple line comment.
   - - - */

Yousaf

Pre-requisite for code:

```
1   #include <iostream>
2   Using namespace std;
3
4   Int main ()
5   {
6
7
8       return 0;
9   }
10
```

line 1)

- # Include is used to call preprocessor directive
- iostream or input-output stream is a preprocessor directive, which means that it has built-in libraries.

line 2)

- std is a library in iostream which we can use for certain functions e.g cout, cin, endl etc.

line 8)

- By running a code, we are giving the code command over the CPU but to return the command we will type return 0.

Day: Date: / /

Without using namespace std:

```
1  # Include  <iostream>
2
3  Int  main()
4  {
5    std :: cout << "programming is fun";
6    ←white→std :: cout << " Hello World!";
7           space ignored by C++
8    return 0;
9  }
10
```

line 5)

:: this is the "Scope resolution operator"

std::cout , this means cout is from scope std.

Escape Sequences:

\n →	newline
\t →	tab (~~8~~ 4 spaces)
\a →	alarm
\b →	back space
\\ →	placing backslash
\r →	carriage return, takes the cursor to start of line and starts over writing.
\' →	for adding single quote.
\" →	for adding double quote

Day: Date:

Idenifiers:

An identifier is a programmer-defined name that represents some elements of a program.

— Simply these are just variable names.

Naming convention of a identifier:

- You can use any name but it should be any of the C++ key words.
- Naming only from C++ alphabets.
- Alpha-numeric values and under score
  (a-z), (A-Z), (0-9) (_)
- It must start from a letter or underscore. Number is not allowed.

Literals:

The data stored in a variable is called a literal. Any value, string, integer, double can be a literal.

Literal = data

example:

1) "Hello World" is a string literal.
2) 55 is a integer literal.
3) 6.5 is a float literal.
4) 'X' is a char literal

Storing data:
Data or literal can be of different sizes e.g 2 byte, 4 bytes 8byte.

- We need to define the amount of space we are giving to a literal, so that no extra space is wasted.

- All types of Data types (int, char etc) have different key words for different siezes

Example: (Integers data types)

short int	2 bytes	-32768 to 32768
unsigned short int	4 bytes	0 to 65,535
int	4 bytes	-2147,483,648 to +ve 21----
unsigned int	4 bytes	
long int	4 bytes	see from book
unsigned long int	4bytes	
Long long int	8 bytes	
unsigned long long int	8 bytes.	

- There are other types of Data too, will read about later. *

Yousaf

Day: Date: / /

Writing a variable:

- This step is called initialization

Initialization

Declaration:

We have to declare what type of variable it is e.g

$\underline{\text{char}}$, $\underline{\text{name}}$;

This is declaring the type

name of variable

Assignment

This is assigning a literal to the variable.

e.g

name = $\underbrace{\text{'A'}}$;

Assigned char name a value of A

char is written in ' ' single quots.

- We can do initialization is a single step: ↓

$\boxed{\text{char name = 'A';}}$

Cin statement:

This means $\underbrace{\text{Console}}_{C}\ \underbrace{\text{input}}_{in}$

We ~~[illegible]~~ requre cin to get an input from the user.

Example:

```
1  #Include <iostream>
2  using namespace std;
3
4  int main()
5  {
6   string name;
7   cout << "Enter your name:\n";
8   cin >> name;
9   cout << "Your name is " << name << endl;
10  return 0;
11 }
```

Terminal

```
Enter your name:
Haider
Your name is   Haider.
```

Line 8:
~~[illegible]~~ We declared a variable name.

Line 8:
cin is used with closing brackets (>>) because we are putting the input into $\xrightarrow{>>}$ name.

Line 9:
---- << name << , the variable cannot be included in the string, so it is written like this.