- Structured (procedual) : Separation of data and functions 10/2
- Object oriented : Encapsulation of data and functions
- Data and programs stored in memory (memory is separated from (cpu)
  Instructions and data are fetched from memory to cpu.
- Machine cycle : (1) Fetch: get instructions from main memory
  (2) Decode: Translate it into computer commands
  (3) Execute: excute the translated commands
  (4) Store: write the result to main memory
- Low level language : (1) The machine language : what computer understand and runs (0's & 1's)
  ↑ compiler (2) Assembly language : set of commands that cpu can understand
- High level language : Everyday English that human can understand.
- cpu → Registers → Accumulator • control unit

13/2

- Software development cycle : Analysis, Design, implementation, testing
- The compilation process : (1) preprocessor: Load the libraries (# include)
  (2) compiler: converts a source code to an object code (0's & 1's)
  (3) Linker: combines all object codes with the required library into (exe)
- Editor (Integrated development environment) : program is created in editor and stored on disk.
- Loader : puts program in memory. • cpu : takes each instruction and excute it
- Single line comment : // → ignore • multiline comment : /* ... */ → scan
- return 0; → return to operating system (success) • int main : is called by the operating system (caller)
- # : directive • include : command for the preprocessor to load a library.
- iostream : Input/output stream library
- # include <filename> : for standard library header files
- # include "filename" : for programmer-defined header files
- using : directive (std: used for cout & cin)
  or using namespace std;
- Scope Resolution operator :: → same name can be in multiple namespaces (prevent ambiguity)
- function : name, ( ), { }, int or float, return 0, indentation (for styling)
- cout : console output

- A variable is named storage location in the computer memory 17/2
- Variable: (1) no spaces (2) 1st character letter or _ (3) no special characters (4) no reserved words
- Decleration: (1) memory allocation (bits) (2) variable name (3) hexa-decimal address
- Type: (1) domain (2) behavior (3) memory
- char is stored in memory as ASCII (ex: 'A': 65 & 'a': 97) . String: "___"
- char: 1 byte int: 4 bytes short int: 2 bytes (↳ range) long int: 4 bytes
- bool: 1 byte float: 4 bytes double: 8 bytes long double: 8 bytes
- Range unsigned (+ve) = 0 to $2^{8n}-1$. Range signed = -2^{8n-1} to $2^{8n-1}-1$
- Sizeof( ) (↳ data type / variable) . data overflow . data Truncation
- whitespace: tab, space, enter ↳ $150-2^8=-106$
- cout << (int) a → print variable a's ascii . print \ , write //
- cout << & a → print variable a's address in memory

20/2

- Type conversion: (1) implicit (automatic): done by compiler, lower-type variable is converted into higher-type (casting / type casting)
  (2) explicit (casting): (type) expression
- constant: (1) #define (2) const → inside main function
- Streams: sequence of bytes (characters)
- cin & cout are two objects (variable of a given type)
- getline (cin, variable) → to input string with spaces
- Setw ( ) → how many spaces . setprecision ( ) → for decimal precision (with d.p) (↳ = / last percision)
- fixed → excluding decimal point . showpoint → to display the decimal point
- oct, hex, dec << variable
- cascading cin / cout: multiple ones . "=" assignment operator (بتنقل الي على يمين فالشمال)

24/2

- A+B ↳ operands
- Ternary operator: Expression 1 ? Expression 2 : Expression 3
- x++ , x+=1 , x = x+1 [equal]
- prefix: ++a : the value of a is increased before the expression is evaluated . "!": unary
- postfix / suffix: a++ : the value of a is increased after the expression is evaluated
- LHS must be a variable (assignment operator) . modulus → العملية على (integers)
- to get average: (1) cast values to float or (2) divide them by float number
- floor (x) → round down . ceil (x) → roundup . trunc → round towards zero

2712

?: → one line statement

switch → used with integer values

( ) ? ( ) : ( ) ;

```
• switch (ch) {
case 'a' : case 'A' : ____ ;
break;
default: ____ ;
}
```

3/3

---

cin.ignore ()

while (expression) action

for (initialization ; expression; expression)
↳ 2 variables
action

• for (; ;) infinite loop

6/3

• use flag or break to exit

• Syntax error: Ex: forgot ; or wrote For not for or Switch (x/1.0)
float

• logical error: Ex: $2^3 = 281$ xx

• runtime error: Ex: $\frac{x}{0}$ البرنامج يتوقف

• Do_while: do {
Statement
} while (condition) ;

Ex: do
Sum += i++;
while (i<n)

• Loop body is executed at least once even if the condition is false

• Break: it will exit from the current loop only

• continue = proceeds with next iteration of loop

- Read data that contain spaces by getline & cin.get (ex: char str [100];) 10/3
- Sizeof: rls هو   . Strlen: rls شال
- Str cat: appends one string at the end of another string   • Void → function malhash return.
- Strcmp: x == y → 0 , x > y → 1, x < y → -1
- Structured programming is a way of building the program by breaking it into small pieces.
- functions are : (1) user - defined (2) built-in (pre-defined)
- Implementing function: • function decleration (prototype): <type> <fun.name> (<type list>); (→ parameter is optional)
  - function call: <function name> (<argument list>)
  - function definition: لو حطناها قبل ال main يبقى مش لازم ال prototype
- Activation record is created for each function : (1) parameters & values (2) location to return

13/3

- End of an array : \0
- Local variables: having life time of the function or block (function and block scope)
- global variables: having life time until the end of program (file level scope) (used in named constants)
- Local variables hide global variables with the same name
- Scope (resolution) operator : :: is used to access a global variable
- Static variables: (1) start (2) scope of local (3) lifetime of global (→ retain value) (4) automatically assigned value 0
- (1) pass - by value: copy the values of the function arguments.
- (2) pass - by reference with reference arguments: (&)
- (3) pass - by refrence with pointer arguments:
  - ↳ refer to the same arguments, any change will make effect, two different names for the same variable
- void <fun name> (int arr[ ] , int <var>) → passing array to function
- arrays are automatically passed by reference (without using &)

17/3

- passing by reference : argument passed to reference parameter must be a variable
- default argument : • must be a constant declared in prototype (or header) and not in definition
  - must be right most & used only with call by value
- function overloading: functions with same name and different parameters
- Recursion: solving a problem by reducing it to smaller versions of itself (function that calls itself)
  - ↳ (1) Base case   (2) general case (rule) → will be reduced to base case (terminate)
  - ↳ may leads to stack overflow