# Progress 1-2 Session 1-2

1 C allows **synthesis (compartmentalization)** of code and data.

2 A **flowchart** is a diagrammatic representation that illustrates the sequence of operations to be performed to arrive at a solution.

3 Flowcharts help us review and debug programs easily. **True**

4 A flowchart can have any number of start and stop points. **False** only 1

5 A **looping** or iterative construct is **basically the execution of a sequence of statements until a particular condition is True or False.**

**Là chuỗi thực hiện các câu lệnh đơn giản cho đến 1 điều kiện chung đúng or sai**

1.C is case sensitive. **(True)** C is case sensitive, do while is different from **DO WHILE**

2.The number 10 is a \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.**whole number**

3.The first character of the identifier can be a number. **(False)**

the first character of the identifier is a letter or an underscore(\_).

4.Using the **float** type saves memory as it takes only half the space as a **double** would.

5.The **void** data type is used to indicate the C compiler that no value is being returned.

6.\_\_\_\_\_\_\_ and \_\_\_\_\_\_\_ are the two classes of arithmetic operators. **Unary and Binary**

7.The unary arithmetic operators are \_\_\_and \_\_\_ . **++ and – –**

The unary arithmetic operators are unary minus (-), increment operator (++) and decrement operator (––).

# Progress 3 - Session 3-4

1 **Operators** are the tools that manipulate data.

2 An **expression** consists of a combination of operators and operands.

3 **Precedence** establishes the hierarchy of one set of operators over another when an expression has to be evaluated.

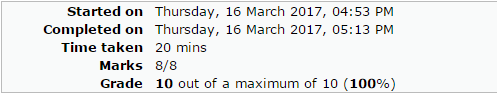
4 **Mixed Mode Expression** is one in which the operands of an operator belong to different data types.

5 An expression can be forced to be of a certain type by using a **cast**

**6 Logical operators** are symbols that are used to combine or negate expressions containing relational operators.

7 Bitwise logical operators are **&, | , ^, ~,etc.**

8 The precedence of operators can be overridden by enclosing the required part of the expression in **Parentheses.**

1. 

# Progress 4 - Session 5-6

1 The formatted I/O functions are **printf()** and **scanf().**

2 scanf() uses \_\_\_\_\_\_\_\_\_ to variables rather than variable names. **pointers**

3 \_\_\_\_\_\_\_\_\_\_\_ specify the form by which the values of the variables are to be input and printed. **format specifier**

4 \_\_\_ is **used by the printf()** function to **identify conversion specifications**. %

5 getchar() is a function without any arguments **True**

**putchar have agruments**

**6** \_\_\_\_\_\_\_\_\_\_\_ is **a temporary storage** area in memory. Buffer

7 Escape sequence can be placed outside the control string in printf().**False**

# Progress 5 - Session 7-8

1 \_\_\_\_\_\_\_\_\_\_ statements enable us to change the flow of a program. **Conditional**

2 The else statement is optional.**True**

3 A \_\_\_\_\_\_\_\_\_\_ is an if statement, which is placed within another if or else. **Nested if**

4 The \_\_\_\_\_\_\_ statement **is a multi-way decision maker** that tests the value of an expression against a list of integer or character constants. **Switch**

5

if (expression)

statement 1

else

statement 2

Which statement will be executed when expression is false? statement 2

# Progress 6 - Session 9-10

1 \_\_\_\_\_\_\_\_\_ allows a set of instructions to be performed until a certain condition is reached. **Loop**

2 \_\_\_\_\_\_\_\_\_ loops check the **condition at the top of the loop** which means the loop code is not executed, if the condition is false at the start. **while**

3 A \_\_\_\_\_\_\_\_\_\_\_ is used to **separate the three parts** of the expression in a for loop. **Semicolon (;)**

4 The \_\_\_\_\_\_\_\_\_ loop checks its condition **at the end of the loop**, that is after the loop has been executed. C. **do..while loop**

5 The \_\_\_\_\_\_\_ statement causes execution **to return to the point** at which the call to the function was made. **return**

6 The \_\_\_\_\_\_\_ statement violates the rules of a **strictly structured** programming language. **goto**

7 The \_\_\_\_\_\_\_\_\_ function causes **immediate termination** of the program and control is transferred back to the operating system. **Exit**

# Progress 7 - Session 11-12

1 An \_\_\_\_\_\_\_\_ **is a collection of data elements** of the same type that are referred by a common name. **Array**

2 **Each member of an array** is identified by the unique \_\_\_\_\_\_\_ or \_\_\_\_\_\_\_ assigned to it. **Index, Subscript**

3 An **array name** and a **variable name** **can be the same**. **False**

4 **Each element of an array** (**cannot)** (fix can) be used where a variable is allowed or required. **False**

5 Two arrays, even if they are of **the same type and size**, cannot be tested for \_\_\_\_\_\_\_\_\_\_. **Equality**

6 String can be defined as a character type array, **which is terminated** by a \_\_\_\_\_ character. **NULL**

7 Arrays can have **more than one dimension.True**

8 The **comparison** of two strings is done with the help of \_\_\_\_\_\_\_ whereas the **interchanging** is done by \_\_\_\_\_\_\_\_. **strcmp,strcpy**

# Progress 8 - Session 13-14

1 A **pointer** provides a way of accessing a variable without referring to the variable directly.

2 Pointers cannot point to arrays. (**False**)

Pointers can point to variables of other fundamental data type variables like int, char, or double or data aggregates like arrays.

3 The **type** of the pointer defines what type of variables the pointer can point to.

4 The two special operators used with pointers are \_\_\_\_ and \_\_\_\_\_. **\* and &**

5 **Addition** and **Subtraction** are the only operations, which can be performed on pointers.

6 Two pointers can be compared only if both these variables are pointing to variables of different types. **(False)**

7 The allocation of memory in this manner, that is, as and when required in a program is known as **Dynamic Memory Allocation** .

# Progress 9- Session 15-16

1 A \_\_\_\_\_\_\_\_\_ is a self-contained program segment that carries out a specific, well defined task. **function**

2 Arguments appearing in the parentheses are termed as \_\_\_\_\_\_\_. **formal parameters** or **formal arguments**

3 If the return is ignored, control passes to the calling program when the closing braces of the code block are encountered. This is termed as \_\_\_\_\_\_\_\_\_\_\_\_.  **calling routine or calling fuction**

4 The function, which calls another function, is known as the \_\_\_\_\_\_\_\_ and the function, which is being called, is known as the \_\_\_\_\_\_\_\_. **calling routine** or **calling function, called routine** or **called function**

5 A \_\_\_\_\_\_\_ is a function declaration that specifies the data types of the arguments. **prototype**

6 \_\_\_\_\_\_\_\_\_ can be referred to only by statements that are inside the code block, which declares them. **local variable**

7 \_\_\_\_\_\_\_\_ are visible to the entire program, and can be used by any piece of code. **global variables**

8 \_\_\_\_\_\_\_\_\_ govern whether one piece of code knows about or has access to another piece of code or data . **scope rules**

9 Arguments are said to be passed \_\_\_\_\_\_\_\_ when the value of the variables are passed to the called function. **using call by value**

10 In\_\_\_\_\_\_\_\_\_, the function is allowed access to the actual memory location of the argument. **call by reference**

**function**

**formal**

**routine**

**calling**

**prototype**

**local**

**global**

**scope**

**value**

**reference**

# Progress 10 - Session 17-18

1 Strings are terminated by the \_\_\_\_\_\_\_\_\_ character. **Null (\0)**

2 The number of characters that can be input into char arr[15] is \_\_\_\_\_\_\_\_\_. **14**

3 Modification of the string pointer can lead to data loss. **True**

(The pointer may be modified to point elsewhere. However, the modification would cause the string to be inaccessible.)

4 The character\_\_\_\_\_ is used to print a new line in printf(). **\n**

5 To use the strcat() function, the \_\_\_\_\_\_\_\_ header file must be included in the program. **string.h**

6 Two pointers can be compared only if both these variables are pointing to variables of different types. **False must be same types**

7 strcmp() returns \_\_\_\_\_\_\_\_\_\_\_ if two strings are identical.identical = one **0**

8 When an array is passed to a function, only its \_\_\_\_\_\_\_\_\_\_\_ is passed. **address**

9 In **call by \_\_\_\_\_\_\_\_\_\_\_\_**, the function is allowed access to the actual memory locations of the arguments. **reference**

# Progress 11 - Session 19

1. A **structure** groups together a number of data items, which need not be of the same data type.

2.Individual structure elements are referenced through the use of the **dot operator**.

3. Values of one structure variable can be assigned to another variable of the same type using a simple assignment statement. **(True)**

4. It is impossible to have one structure within another structure. **(False)**

5. A new data type name can be defined by using the **typedef** keyword.

6. In bubble sort, the **adjacent ( liền kề )** elements are compared.

7. In insertion sort, if an unsorted element has to be put in a particular sorted location, values are swapped. **(False)**

# Progress 12 - Session 21

1 The two types of streams are the **text** and **binary** streams.

2 Open files are closed when a program crashes. **(False)**

3 The **fopen** function opens a stream for use and links a file with that stream.

4 The function used for writing characters to a file is **fputc**.

5 The fgets() function considers a new line character as a part of the string. **(True)**

6 The **rewind** function resets the file position indicator to the beginning of the file.

7 Whenever a character is read from or written to the stream, the **current active pointer** is incremented.

8 Files on which fread() and fwrite() operate must be opened in **binary** mode.

9 The current location of the current active pointer can be found with the help of the **ftell** function.