DAILY ASSESSMENT FORMAT

|  |  |  |  |
| --- | --- | --- | --- |
| Date: | 18th June 2020 | Name: | Persis P |
| Course: | Solo Learning | USN: | 4AL17EC069 |
| Topic: | Basic Concepts, Conditionals and  Loops, Functions, Arrays &  Pointers, Strings and Function  Pointers | Semester & Section: | 6th sem B sec |
| Github  Repository: |  |  |  |

|  |
| --- |
| FORENOON SESSION DETAILS |
| Image of session |

|  |
| --- |
| Report –  Introducing C  C is a general-purpose programming language that has been around for nearly 50 years.  C has been used to write everything from operating systems (including Windows and many others) to complex programs like the Python interpreter, Git, Oracle database, and more.  The versatility of C is by design. It is a low-level language that relates closely to the way machines work while still being easy to learn.    Hello World!  Let's break down the code to understand each line:  #include <stdio.h> The function used for generating output is defined in stdio.h. In order to use the printf function, we need to first include the required file, also called a header file.    int main() The main() function is the entry point to a program. Curly brackets { } indicate the beginning and end of a function (also called a code block). The statements inside the brackets determine what the function does when executed.    #include <stdio.h>    int main() {  printf("Hello, World!\n");  return 0;  }  Data Types  C supports the following basic data types: int: integer, a whole number. float: floating point, a number with a fractional part. double: double-precision floating point value. char: single character.    The amount of storage required for each of these types varies by platform.  C has a built-in sizeof operator that gives the memory requirements for a particular data type.  Variables  A variable is a name for an area in memory.  The name of a variable (also called the identifier) must begin with either a letter or an underscore and can be composed of letters, digits, and the underscore character.  Variable naming conventions differ, however using lowercase letters with an underscore to separate words is common (snake\_case).  Variables must also be declared as a data type before they are used.  Formatted Input  The scanf() function is used to assign input to variables. A call to this function scans input according to format specifiers that convert input as necessary.  If input can't be converted, then the assignment isn't made.  The scanf() statement waits for input and then makes assignments:  int    x;    float    num;    char    text[20]  ;    scanf  (  "%d %f %s", &x, &num, text);      Typing 10 22.5 abcd and then pressing Enter assigns 10 to x, 22.5 to num, and abcd to text.  Note that the & must be used to access the variable addresses. The & isn't needed for a string because a string name acts as a pointer.    Comments  Comments are explanatory information that you can include in a program to benefit the reader of your code. The compiler ignores comments, so they have no affect on a program.  A comment starts with a slash asterisk /\* and ends with an asterisk slash \*/ and can be anywhere in your code.  Comments can be on the same line as a statement, or they can span several lines.  For example:  #include <stdio.h>    /\* A simple C progra  m    \*  Version 1.0    \*/    int    main() {      /\* Output a  string    \*/      printf  (  "  Hello World!");      return 0  ;    }    Si  ngle  -  line Comments    C++ introduced a double slash comment // as a way to comment single lines. Some C compilers also support this comment style. For example:  #include  <  stdio.h  >      int    main() {      int    x = 42; //  int    for a whole number      For example, the following program evaluates the    expression and then executes the else clause  statement:    #include  >  stdio.h  <      int    main() {      int    score = 89  ;        if (score >= 90  )      printf  (  "Top 10%%.  \  n");      else      printf  (  90.  "Less than  \  n");          return 0  ;    }      Conditional Expressions    Another way to form an if  -  else statement is by using the ?: operator in a conditional expression. The  ?: operator can have only one statement associated with the if and the else.    For example:    #include  >  stdio.h  <      int    main() {      int    y;      int    x = 3;        y = (x >  =  5) ? 5 : x;      /\* This is equivalent to  :      if (x >  =    5)      y = 5;      else      y = x;    \*/        ;  return 0    }      The switch Statement    The switch statement branches program control by matching the result of an expression with a  constant case value.      The switch statement often provides a more elegant solution to if  -  else if and nested if statements.    The switch takes the form:    switch (expression) {      case val1: |

A function:

• is a block of code that performs a specific task

• is reusable

• makes a program easier to test

• can be modified without changing the calling program

Even a simple program is easier to understand when

main() is broken down into subtasks that are

implemented with functions.

For example, it's clear that the goal of this program is to calculate the square of a number:

int

main() {

int

x, result;

x = 5;

;

result = square(x)

printf

"%d squared is %d

(

\

n"

, x, result);

;

return 0

}

Recursive Functions

An algorithm for solving a problem may be best implemented using a process called recursion.

Consider the factorial of a number, which is commonly written as 5! = 5 \* 4 \* 3 \* 2 \* 1.

This calculation can al

so be thought of as repeatedly calculating num \* (num

-

1.

1)

until num is

A recursive function is one that calls itself and includes a base case, or exit condition, for ending the

recursive calls. In the case of computing a factorial, the base case is num

equal to 1.

For example:

#include

>

stdio.h

<

//function declaration

int

factorial(

int

num);

int

)

{

main(

int

x = 5;

printf

factorial of %d is %d

(

"T

he

\

n", x, factorial(x))

;

return 0

;

}

//function d

efinition

int

(

factorial

int

num) {

if (num ==

1) /\* base case \*/

urn (1)

ret

;

else

return (num \* factorial(num

-

1))

;

}