Now test yourself: answers

Unit 4: Digital development concepts

Page 82

1. In procedural programming, the program operates on data and is organised into self-contained blocks called procedures. The logic of the program is actioned by calling procedures. In object-oriented programs the focus is on the data rather than the processes. The data and the methods are contained in a single object. Classes are used to define objects. Classes make use of inheritance.

2. Any three from:
   - Editing window
   - Clipboard
   - Colour coded syntax
   - Collapsible code sections
   - Line numbering
   - Code completion tools

3. The compiler performs a number of passes. The first pass syntax is checked and errors are corrected by the programmer. Library code is added. The compiler converts source code to binary code and checks that statements conform to the grammar rules of the language. The compiler checks to ensure that all variables are declared.

Page 90

1. 10111, 110101, 1100100, 1000000

2. Both are standards on how to represent different characters in binary so that they can be stored, transmitted, and read in digital form. ASCII originally used seven bits to encode each character. This was increased to eight with Extended ASCII to allow for more character representation. Unicode 32, 16, and 8-bit encodings. Using more bits allows more characters to be represented.

3. Any two from:
   - Sign and magnitude: the left-most bit is reserved for the sign and the rest of the binary pattern represents the size.
   - One’s complement: each bit is inverted to create a negative representation.
   - Two’s complement: change the bit pattern to one’s complement add 1. The left-most bit is 1, this represents a negative number.

4. 00011001 + 00001001 = 00100010
   01100100 + 00011100 = 10000000
   00100010 – 00010001 = 10001
   01011000 + 00110010 = 10001010

No overflow. 255 is the largest number that can be represented in eight bits.
5.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D=NOT C</th>
<th>E=A AND D</th>
<th>F=D OR E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Page 104
2. Bubble sort – the algorithm works by swapping adjacent data items until they are in the correct order. Insertion sort – creates a sorted sublist and works on unsorted items. The bubble sort is less efficient for sorting large amounts of data as the time taken to do so is related to the square of the number of items. The insertion sort is adaptive; it adapts to the order of the elements in the list. It is a more stable sort method.
3. Linear search – all items in the list are examined in the order in which they appear. Binary search – works on a sorted list of data. The search will compare data items at the midpoint in the data. If the search value is not found the list is chopped at the midpoint. This continues until the search value is found or the last element in the list is reached.
4. (a)
   1. FOR EACH EMPLOYEE (1 TO 10)
   2. DO
   3. Valid=TRUE
   4. OUTPUT ‘ENTER WAGES’
   5. GET wages
   6. IF wages < 0 OR wages > 200 THEN
   7. Valid=FALSE
   8. OUTPUT ‘Error – re-enter wages’
   9. ENDF
   10. WHILE (Valid=FALSE)
   11. TOTALPAY=TOTALPAY+wages
   12. ENDFOR
   13. AVERAGE=TOTALPAY/10
   14. OUTPUT AVERAGE
(b)

<table>
<thead>
<tr>
<th>LINENO</th>
<th>Valid</th>
<th>Wages</th>
<th>TOTALPAY</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>TRUE</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>TRUE</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>TRUE</td>
<td>250</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>TRUE</td>
<td>250</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>FALSE</td>
<td>250</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>FALSE</td>
<td>250</td>
<td>0</td>
<td>Error – re-enter wages</td>
</tr>
<tr>
<td>9.</td>
<td>FALSE</td>
<td>250</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>FALSE</td>
<td>250</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>FALSE</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>TRUE</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>TRUE</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>TRUE</td>
<td>100</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>TRUE</td>
<td>100</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>TRUE</td>
<td>100</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>TRUE</td>
<td>100</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>TRUE</td>
<td>100</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>TRUE</td>
<td>100</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>TRUE</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>TRUE</td>
<td>100</td>
<td>100</td>
<td></td>
</tr>
<tr>
<td>etc</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(c) Any two from:
- Replace for loop with a while loop to allow for more than 10 entries.
- Improve error message to help user.
- Improve screen prompts to include the number of wages entered e.g. ‘Enter wages for employee 1’ etc.

Page 116
1. (a) Variable – a named location in memory used to hold data when a program is running. Constant – a named location in memory used to hold data whose value does not change while a program is running. Boolean – a data type which can take on only two values, true or false. It can be used to combine conditions and evaluate their outcome.
(b)
1. DO
2. Valid=TRUE
3. OUTPUT “ENTER MARK : “
4. GET mark
5. IF mark < 0 OR mark>100 THEN
6. Valid=FALSE
7. OUTPUT “Error – re-enter mark”
8. ENDIF
9. WHILE (Valid=FALSE)
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] marks = new int[10];
            int temp;
            bool valid = true;

            for (int i = 0; i < 10; i++)
            {
                do
                {
                    valid = true;
                    Console.WriteLine("Enter marks ");
                    marks[i] = Convert.ToInt32(Console.ReadLine());
                    if (marks[i] < 0 || marks[i] > 100)
                    {
                        valid = false;
                        Console.WriteLine("Marks must be in the range 1-100");
                    }
                }
                while (valid == false);
            }

            for (int i = 0; i < marks.Length - 1; i++)
            {
                for (int j = 0; j < marks.Length - 1 - i; j++)
                {
                    if (marks[j] > marks[j + 1])
                    {
                        temp = marks[j];
                        marks[j] = marks[j + 1];
                        marks[j + 1] = temp;
                    }
                }
            }
            Console.WriteLine("Sorted List of Marks");
            for (int i = 0; i < marks.Length; i++)
            {
                Console.Write(marks[i] + " ");
            }
            Console.ReadKey();
        }
    }
}
2. Linear search – all items in the list are examined in the order in which they appear. Binary search – works on a sorted list of data. The search will compare data items at the mid point in the data. If the search value is not found the list is chopped at the midpoint. This continues until the search value is found or the last element in the list is reached. The binary search is more efficient because it carries out less comparison and uses a data set that decreases in size when searching for the search value.

3. (a) Any three from:
   - Functions can be re-used.
   - Functions can be tested and debugged individually.
   - Functions allow programmers to structure code so that it is more efficient.
   - Data can be passed to functions.
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int[] marks = new int[10];
            int temp;
            bool valid = true;

            for (int i = 0; i < 10; i++)
            {
                do
                {
                    valid = true;
                    Console.WriteLine("Enter marks ");
                    marks[i] = Convert.ToInt32(Console.ReadLine());
                    if (marks[i] < 0 || marks[i] > 100)
                    {
                        valid = false;
                        Console.WriteLine("Marks must be in the range 1-100");
                    }
                }
                while (valid == false);
            }

            for (int i = 0; i < marks.Length - 1; i++)
            {
                for (int j = 0; j < marks.Length - 1 - i; j++)
                {
                    if (marks[j] > marks[j + 1])
                    {
                        temp = marks[j];
                        marks[j] = marks[j + 1];
                        marks[j + 1] = temp;
                    }
                }
            }

            outputData(marks);
        }

        static void outputData(int[] marks)
        {
            Console.WriteLine("Sorted List of Marks");
            for (int i = 0; i < marks.Length; i++)
                Console.Write(marks[i] + " ");
            Console.WriteLine();
            Console.ReadKey();
        }
    }
}
(c) Files store data permanently. Programming languages need to use files if data is to be stored after the program has stopped running. The data in a file can be used as input to the program.

**Page 118**

1. Data validation ensures that the data entered is complete, falls within a set of boundaries specified and is sensible. It aims to reduce the data entry errors.

2. Presence check – ensures a field on a form contains data, e.g. a form may use a presence check to ensure an email address has been entered.

   Length check – ensures a field on a form contains a set number of characters, e.g. a date of birth recorded in the format ‘08/08/05’ must contain 8 characters.

   Range check – ensures a field on a form contains data in a given range, e.g. marks for an exam must be in the range 1–100.

   Type check – ensures that a field on a form contains data of a specified type, e.g. a customer number field can contain numbers only.

3. Syntax errors occur when the program is being edited. The program will not run if syntax errors are present. Execution errors are detected when the program is running, for example if the program attempts to divide by zero.

4. Error trapping can be carried out using the debug tool. Breakpoints are set in the program and variables and their values can be inspected for correctness. Exception handling can be used to stop a program from crashing. If an error occurs the program calls the error handling code and the user receives a message. The program will continue to run.

**Page 122**

1. Aims to improve the quality of software by ensuring that it is bug free, meets the user requirements and operates efficiently.

2. White box testing – tests the internal logic of a section of code to identify syntax, logic and data flow errors. It should be built so that all statements are executed at least once.
   - **Advantages:** Each line of code is tested; hidden errors are identified and the code is modified.
   - **Disadvantages:** Experienced programmers are required; large scale applications require complex test cases.

Black box testing – focuses on testing inputs and outputs. The unit of code is viewed as a black box. It is used to identify errors in data structures/problems with user inputs/missing functionality. The tester has no knowledge of the internal code.
   - **Advantages:** can help identify where the user requirements are not being met; testers do not need to know about the code.
   - **Disadvantages:** not all program pathways are tested; some tests carried out by white box testers are repeated.
3. (a) Unit testing – a unit of code is tested in isolation to ensure it is working; it is a form of white box testing. Integration testing – occurs after unit testing; a number of tested units are combined together to form a subsystem, these are tested to ensure they work together as expected. System testing – occurs after integration testing; several subsystems are brought together and tested; a form of black box testing.

(b) Any three from:
- Bugs or errors in the code
- Missing functionality
- Incorrect calculations or incorrect output
- Inefficiencies in code

4. (a) Test number, reason for test, test data, expected output, observed output, whether observed matches expected.

(b) Valid data is data that falls within the specified boundaries. This is used to ensure that the system will work as expected using normal data.
Invalid data is data that is not acceptable. It is used to ensure that the system will reject invalid values and continue to operate.
Null data is used to ensure that the system can cope when no data is entered in a particular field.
Extreme data is data that lies at the boundaries, e.g. 0–100 for a percentage. It is used to ensure that the system can cope with values at the two boundaries.

Page 124
1. A set of tasks that the software should perform in order to solve the problem described by the user. The list is created after consultation with the user. The user requirements form the basis for evaluating software.
2. Evaluation is ongoing throughout the development process. Both the user and the programmer are involved in evaluation. Evaluation involves testing and reviewing the system to ensure it meets the user requirements. Evaluation may lead to change and improvement.
3. An evaluation report contains purpose of evaluation, the timing and stage of development of the evaluation, the outcome of previous evaluations, name and roles of those involved.
4. System robustness, system performance, possible improvements, deviations from original design, strengths and weaknesses.
5. The test plan can reference the user requirements and can determine the extent to which the system meets these requirements. It can plan for robustness testing by including a suitable range of tests. It can assess how well the system works with normal data. Tests for usability and performance can also be created.
6. The ability of a system to handle valid, invalid and exceptional data. If the system does not crash when processing high volumes of this type of data it is said to be robust.