

Course Prefix/Number: CPT 168
Course Title: Programming Logic and Design
Lec. Hours/Week: 3.0
Lab Hours/Week: 0.0
Credit Hours/Semester: 3.0

[Distance Learning Attendance/VA Statement](#)
[Textbook Information](#)

COURSE DESCRIPTION

This course examines problem-solving techniques applied to program design. Topics include a variety of documentation techniques as means of solution presentation.

COURSE COMPETENCIES

Upon successful completion of this course, the student should be competent to complete the following tasks:

Module 1 – Introduction to Problem Solving with Computer Programming

- Define algorithmic and heuristic solutions
- Describe the difference between constants and variables
- Define common data types used in most high-level programming languages: integer, real, logical, character, string
- Choose a data type based on the way a variable will be used
- Evaluate mathematical expression using the order of operations using arithmetic operators
- Evaluate relational expressions using relational operators
- Define the logical operators: NOT, AND, OR
- Construct truth tables for given logical expressions
- Define variables and construct a logical expression to corresponds with a given logical statement

Module 2 – Modular Program Development

- Describe the benefits of modular program development
- Explain the concepts of cohesion and coupling
- Employ global and local variables as necessary
- Employ call-by-value and call-by-reference as necessary
- Develop a module hierarchy chart for a program that uses several modules
- Employ flowchart symbols to visualize the data flow through a program
- Trace through the program providing all output and variable values given as algorithm and/or flowchart
- Relate the need for internal and external program documentation

Module 3 – Control Structures

- Recognize the flowchart structure for algorithms using sequential, decision, loop, or case logic
- Create flowchart and algorithm for a solution to a problem requiring the use of decisions
- Create the flowchart, eliminate unnecessary conditions, and construct the corresponding algorithm given a decision table.
- Relate the differences between major loop structures
- Define and use counters and accumulators
- Choose the correct loop structure to solve a given problem
- Create a solution to a problem requiring nested loops
- Create the flowchart and algorithm for a solution that uses the case logic structure

Module 4 – Arrays and File Access

- Relate the benefits of using arrays in programming
- Describe the difference between static and dynamic arrays
- Solve problems requiring input, output, and manipulation of array values

- Solve problems using parallel arrays
- Solve problems using the pointer technique
- Describe the linear search algorithm
- Describe the binary search algorithm
- Solve a problem using a multi-subscripted array
- Describe the difference between sequential and direct file access
- Create solutions to problems that involve file input and output.

MINIMAL STANDARDS

Minimal standards of performance on all course competencies for receiving credit for the course and indicated by 60% overall accuracy on evaluation instruments that address the course competencies listed above. Required standards of performance on all course competencies for enrollment in subsequent higher-level computer technology courses are indicated by 70% overall accuracy on evaluation instruments that address the course competencies listed above.

COURSE REQUIREMENTS

Students are responsible for attending all schedule class meetings until they have completed all course requirements. Students are responsible for all material covered and for all assignments made in all classes. Any student caught cheating or involved in other academic dishonesty will be given a grade of zero and will be subject to further disciplinary action.

Attendance Policy

The attendance policy as stated in the York Technical College Handbook will be enforced. Make-up tests will not be given for theory tests. If a student must miss a theory test, he/she will get a zero for that test. However, students have the option of taking the comprehensive final. The student's grade on the comprehensive final will replace his/her lowest theory test grade. It is the student's responsibility to schedule a time for a make-up hands-on test with his/her instructor.

Evaluation Strategies/Grading

Module 1 (22.5% total)	Module 2 (22.5% total)	Grading Scale	
		Tests – 12.5% of final average Homework – 10% of final average	Test(s) – 12.5% of final average Program(s) – 7.5% of final average Homework – 2.5% of final average
Module 3 (22.5% total)	Module 4 (32.5% total)	89-89	B
		70-79	C
		60-69	D
Test(s) – 12.5% of final average Program(s) – 7.5% of final average Homework – 2.5% of final average	Test(s) – 12.5% of final average Program(s) – 15% of final average Homework – 5% of final average	Below 60	F

Entry Level Skills

A student entering this course should have a solid foundation in algebra.

Prerequisites: MAT 101

Disabilities Statement: Any student who feels s/he may need an accommodation based on the impact of a disability should contact the Special Resources Office (SRO) at 803-327-8007 in the 300 area of Student Services. The SRO coordinates reasonable accommodations for students with documented disabilities.

TOPIC/CONTENT OUTLINE

Module 1

Introduction to Problem Solving with Computer Programming

- A. Defining algorithmic and heuristic solutions
- B. Describing the difference between constants and variables
- C. Defining common data types used in most high-level programming languages; integer, real, logical, character, string
- D. Choosing a data type based on the way a variable will be used
- E. Using arithmetic operators to evaluate mathematical expressions using the order of operations
- F. Using relational operators to evaluate relational expressions
- G. Defining the logical operators: NOT, AND, OR
- H. Constructing truth tables for given logical expressions
- I. Defining variables and constructing logical expressions to correspond with given logical statements

Module 2

Modular Program Development

- A. Describing the benefits of modular program development
- B. Explaining the concepts of cohesion and coupling
- C. Using global and local variables
- D. Using call-by-value and call-by-reference
- E. Developing hierarchy charts for programs involving several modules
- F. Using flowchart symbols to visualize the data flow through a program
- G. Tracing through algorithms and flowcharts in order to provide all output and variable values
- H. Relating the need for internal and external program documentation

Module 3

Control Structures

- A. Recognizing the flowchart structure for algorithms involving sequential, decision, loop, and case logic
- B. Creating flowcharts and algorithms for solutions to problems requiring the use of decisions
- C. Creating flowcharts, eliminating conditions, and constructing algorithms for given decision tables
- D. Relating the difference between major loop structures
- E. Defining and using counters and accumulators
- F. Choosing the best loop structure to solve a given problem
- G. Creating solutions to problems that require nested loops
- H. Creating flowcharts and algorithms for solutions that use the case logic structure

Module 4

Arrays and File Access

- A. Relating the benefits of using arrays in programming
- B. Describing the difference between static and dynamic arrays
- C. Solving problems that require input, output, and manipulation of array values
- D. Solving problems using parallel arrays
- E. Solving problems using the pointer technique
- F. Describing the linear search algorithm
- G. Describing the binary search algorithm
- H. Solving problems using a multi-subscripted arrays
- I. Describing the difference between sequential and direct file access
- J. Creating solutions to problems that involve file input and output