

Public and Catholic District School Board Writing Partnerships

Course Profile

Computer and Information Science

Grade 11

University/College Preparation

ICS3M

• *for teachers by teachers*

This sample course of study was prepared for teachers to use in meeting local classroom needs, as appropriate. This is not a mandated approach to the teaching of the course. It may be used in its entirety, in part, or adapted.

Course Profiles are professional development materials designed to help teachers implement the new Grade 11 secondary school curriculum. These materials were created by writing partnerships of school boards and subject associations. The development of these resources was funded by the Ontario Ministry of Education. This document reflects the views of the developers and not necessarily those of the Ministry. Permission is given to reproduce these materials for any purpose except profit. Teachers are also encouraged to amend, revise, edit, cut, paste, and otherwise adapt this material for educational purposes.

Any references in this document to particular commercial resources, learning materials, equipment, or technology reflect only the opinions of the writers of this sample Course Profile, and do not reflect any official endorsement by the Ministry of Education or by the Partnership of School Boards that supported the production of the document.

© Queen's Printer for Ontario, 2001

Acknowledgments

This profile was a collaborative effort of the Institute for Catholic Education (ICE) and the Halton District School Board.

Public School Board Writing Team - Grade 11 Computer and Information Science
Lead Board

Halton District School Board
Hans van Wijk , Project Manager

Course Profile Writing Team - Public

Mark Richardson, Halton District School Board (Lead Writer)
Jaye Herbert, Thames Valley District School Board
Dan Visentin, Halton District School Board

Local Reviewers

Angela Elksnitis, Mohawk College
Derek Murphy - Industry

Catholic School Board Writing Team - Grade 11 Computer and Information Science
Lead Board

Dufferin-Peel Catholic District School Board
Denise Panunte, Project Manager

Course Profile Writing Team - Catholic

Roy Parteno, Dufferin-Peel Catholic District School Board (Lead Writer)
Kirstine Fenwick, Dufferin-Peel Catholic District School Board
Greg Rodrigo, Georgian College (formerly of Dufferin-Peel)

Local Reviewers

Sandy Graham, University of Waterloo
Rosaria Kalino, Dufferin- Peel Catholic District School Board
Carmen Leith Dufferin- Peel Catholic District School Board (retired)
Chris Stephenson, Association of Computer Studies Educators

Course Overview

Computer and Information Science, University/College Preparation, ICS3M

Course Description

This course helps students examine computer science concepts. Students outline stages in software development, define standard control and data structures, identify on- and off-line resources, explain the functions of basic computer components, and develop programming and problem-solving skills using operating systems and implementing defined practices. As well as identifying careers in computer science, students develop an understanding of the ethical use of computers and the impact of emergent technologies on society.

How This Course Supports the Ontario Catholic School Graduate Expectations

The Computer and Information Science program in the Catholic faith community enables young adults to develop and utilize their gifts and resources in finding solutions that benefit others in ways that model Gospel values. The curriculum focus enables students to be critical thinkers and innovative problem solvers and analyse the use of resources while understanding the implications of technological innovations. Emphasis on process and results ensures students apply skills and knowledge when providing services and recognize our God-given responsibility to respect the dignity and value of the individual and the protection of the environment. Computer technology has an ever-increasing effect upon society (e.g., the digital divide: the division of groups in society based upon the access to information that further disadvantages the poor). It is important for young Catholics to reflect upon and examine the potential of technology to affect lives.

Course Notes

Over the past ten years, there has been an effort to increase enrollment in computer-related programs at the postsecondary level. This course prepares students for further study at the Grade 12 College and University levels. The combination of theory, practical experience, and exploration of career options also helps students complete and refine their Annual Education Plans.

The Computer and Information Science Grade 11 course prepares students for College and University destinations. Students outline opportunities and career paths of each destination in their Annual Education Plans. They also explore career paths and identify which career best suits their interests, aptitudes, and expectations.

This course has no prerequisite. Some students will have completed Grade 10 Computer and Information Science or Computer Engineering and have been introduced to basic programming concepts and structures. For others, it is the first Computer and Information Science course. The focus of this course is on applying fundamental programming structures and concepts rather than applying a specific language and its features. Upon completion of this course, students can apply their knowledge and skills to other programming languages.

Problem solving, a curriculum sub-heading, is integrated in all units. The software design life cycle is followed when developing the best solution to a challenge.

Students often research and use the Internet as an information source. It is important for teachers to review and emphasize good information filtering skills. A session with the school teacher-librarian may assist all students.

Challenges are drawn from a variety of disciplines and workplace situations. They address the wide spectrum of student interests, provide opportunities for broad applications of programming, and are free of bias.

Communication is a key skill of programmers. In addition to reading and writing, programmers communicate using programming standards and conventions, and through developing internal and external documentation. These skills are integral to every unit.

The final unit is an authentic assessment in which students apply a wide range of knowledge and skills through an integrated and meaningful task. This task is a program challenge in which the software design life cycle (problem definition, analysis, design, implementation, testing, maintenance) is followed as the problem-solving model.

Students develop generic programming skills in this course. When choosing a programming language, the following criteria are applied:

- ease-of-use, appropriate structure, availability, and hardware requirements;
- level of difficulty allowing students without programming background to experience success;
- provision of de-bugging tools;
- planned path for language skill development in Grades 11 and 12, considering the most likely postsecondary destination and required preparation within the school community;
- district standards, conventions, and policy;
- available on- and off-line support resources;
- compatibility with languages used in introductory courses at local postsecondary destinations.

Units: Titles and Times

Unit 1	Working in the Computing Environment	12 hours
Unit 2	Beginning to Program	25 hours
* Unit 3	Problem Solving with Procedures and Functions	18 hours
Unit 4	Information Storage and Related Issues	12 hours
* Unit 5	Using Data Structures	18 hours
Unit 6	Putting It All Together	25 hours
	TOTAL	110 hours

* These units are fully developed in this Course Profile.

Unit Overviews

Unit 1: Working in the Computing Environment

Time: 12 hours

Unit Description

This unit focuses on basic computer and information science skills. Students identify hardware components, research ergonomic considerations, practise file management skills, access resources through local and wide area networks, and research the evolution of programming languages. They develop skills for success in the computer and information science environment. Students focus on the Computer and Information Science environment; students also examine respect for the environment and wise use of resources from a Catholic perspective.

Unit Overview Chart

Activity	Expectations	Assessment	Focus
1	TFV.05, TF3.01, TF3.02; CGE7i	C; K/U	What's in the Lab?
2	TF3.02, ICV.01, ICV.02, IC1.01, IC1.05; CGE1d, 7i	K/U; C; A; T/I	Comfortable Computing
3	TF3.03, SPV.04, SP3.01, SP3.02, SP3.05; CGE2b	C; A	Computer Survival Skills
4	SP1.01, SP1.08, SP2.08, SP3.03, SP3.04, IC1.03; CGE2b, 4f	T/I; A; K/U	Information Management Case Study
5	TF2.11, TF2.15; CGE7g	C; K/U	History of Programming

K/U = Knowledge/Understanding

C = Communication

T/I = Thinking/Inquiry

A = Application

Unit 2: Beginning to Program

Time: 25 hours

Unit Description

This unit focuses on basic programming structures. Students write simple programs, using variable assignment, repetition, and decision structures, and develop effective testing, validating, and documenting skills. They also explore roles of effective communicators and reflective thinkers when following a problem-solving model (e.g., user inputs a series of marks, each value is validated, the average is calculated, and a grade is assigned).

Unit Overview Chart

Activity	Expectations	Assessment	Focus
1	TF1.01, TF2.01, TF2.02, TF2.10, SPV.01, SP2.01, SP2.11, SP2.13, SP2.15; CGE2f	K/U; C; A; T/I	Input and Data Storage in Simple Programs
2	TF2.02, TF2.03, TF2.10, SPV.02, SP1.02, SP2.09, SP2.10, SP2.13; CGE3c	K/U; C; A; T/I	IPO (Input, Processing, Output) in Program Format
3	TF2.06, SP2.04, SP2.07; CGE3c	C; A	Introducing Selection
4	TF1.07, TF2.06, TF2.14, SP2.04, SP2.07C; GE4b	C; A	Introducing Repetition
5	TF2.04, SP1.05, SP2.04, SP2.07; CGE3c	C; A; T/I	Combining Selection and Repetition
6	TF1.05, SPV.02, SP2.12, SP2.14; CGE4f	K/U; A; T/I	Program De-bugging, Verification, and Documentation
7	TF2.04, SP1.05, SP2.04, SP2.07; CGE4d, 5a	C; A	Team Programming Project

Unit 3: Problem Solving with Procedures and Functions

Time: 18 hours

Unit Description

This unit focuses on program modularity and career exploration. Students write programs using existing sub-programs and then progress to writing programs including their own sub-programs. They also explore careers in computer studies and develop skills in program modularity (e.g., a program to encrypt/decrypt a passage of text using substitution encoding). Students complete a reflection on work and on the *Laborem Exercens* encyclical.

Unit Overview Chart

Activity	Expectations	Assessment	Focus
1	TFV.02, TF1.03; CGE7a	K	The Software Development Process
2	TF2.07, SP2.05; CGE2b	K; I	Investigating Math and Text Subroutines
3	SP2.06, SP1.09, TF2.08, TF2.09	K; I; A	Defining Our Own Subroutines
4	ICV.03, IC1.06, IC2.01, IC2.02, IC2.03; CGE5b	K; I	Exploring Careers in Computing
5	SPV.01, SP2.06; CGE2b, 5a	I; C; A	Programming with Subroutines
6	CGE1g, 2b, 4g, 5b, 5d	K; I; C; A	Reflecting on Work

Unit 4: Information Storage and Related Issues

Time: 12 hours

Unit Description

This unit focuses on data storage and manipulation. Students examine issues surrounding privacy, security, and ethical use of information. They also write programs that input data from existing files, process the data, and create files for external data storage, following an appropriate problem-solving model (e.g., Create a data file containing employee information including hours worked and rate of pay. Read from the file, compute, display, and write to a new file the gross pay for each employee.).

Unit Overview Chart

Activity	Expectations	Assessment	Focus
1	IC1.02; CGE1d, 7e	C	Information Impact
2	TFV.01, TF1.02, TF1.04; CGE2b	C; T/I	Choosing Your Tools
3	SP2.17, SP2.18; CGE1d, 2e	A	Reading Data Files
4	SP2.17, SP2.18; CGE1d, 2e	A	Creating Data Files

Unit 5: Using Data Structures

Time: 18 hours

Unit Description

This unit focuses on the programming techniques required to store and manipulate data and to solve problems through the development of a database. Each activity develops knowledge and skills that students apply in the culminating challenge of this unit: to develop a database for a school team (e.g., the hockey team or similar organization, consisting of personal data such as player name, position played, jersey number, phone number, goals, and assists). Students examine the structuring of one- and two-dimensional arrays and how data is represented and stored in these structures. They write programs that create lists and tables of data, manipulate the data, and output the result. Sorting and searching techniques are also applied.

Unit Overview Chart

Activity	Expectations	Assessment	Focus
1	TFV.03, TF2.05; CGE4f	C; K/U	Examining Data Structures
2	SP1.03, SP2.02, SP2.10, SP2.14, SP2.15, SP2.16; CGE7h	K/U; A	Data in Lists
3	SP1.03, SP2.02, SP2.03, SP2.10, SP2.14, SP2.15; CGE5a	K/U; A	Relating Lists
4	TFV.03, TF2.05, SP1.03, SP2.02, SP2.10, SP2.14, SP2.15; CGE5e	K/U; A	Data in Tables
5	SP1.07, SP2.02, SP2.10, SP2.14, SP2.15, SP2.16; CGE3c	A	Sorting Data
6	SP2.02, SP2.10, SP2.14, SP2.15, SP2.16; CGE7h	A	Searching Lists and Tables

Unit 6: Putting It All Together

Time: 25 hours

Unit Description

This unit is the culminating challenge for applying knowledge and skills in an integrated and meaningful task. Students follow the software design life cycle to find the best solution to a challenge (e.g., a movie reservation system), demonstrating the mastery of course expectations. The teacher should choose the challenge with students to allow students to express their creativity while at the same time demonstrating knowledge and skills. Students examine the effect and influence on society of emergent technologies.

Unit Overview Chart

Activity	Expectations	Assessment	Focus
1	SP1.04; CGE3b	A	Defining the Problem
2	TFV.04, TF1.06, TF2.13, SP1.04; CGE3c,4f	K/U; A; T/I	Developing a Plan
3	TF2.12, SPV.03, SP1.04; CGE3c	A; T/I	Creating a Solution
4	SPV.03, SP1.04; CGE4d	A	Evaluating the Solution
5	SPV.03, IC1.04; CGE2c	A; C	Communicating the Results

Teaching/Learning Strategies

Teaching a course in a computer lab is a unique experience as compared to teaching in a classroom. Teaching strategies should include plans to balance computer usage and group instruction and means to make sure that attention is paid to the discussion (e.g., turning computer monitors off during discussions). A variety of teaching and learning strategies are used, including:

Brainstorming: expressing initial ideas with neither criticism nor analysis, e.g., problem-solving discussion in the problem definition and analysis phases of the software life cycle;

Collaborative/Cooperative: small-group learning providing high levels of engagement and interdependence (e.g., students working as a team to develop components of a computer program);

Conferencing: student-to-student discussion;

Software Life Cycle Design Process: problem-solving approach using a prescribed series of steps;

Computer-based Tutorials/Exploration Activities: use of installed and networked resources, open-ended explorations, and computer projectors, allowing students to work as the teacher demonstrates;

Independent Study: exploring and researching a topic of interest;

Programming: developing software solutions;
Computer Research: using on- and off-line resources;
Report/Presentation: presenting research topics to the class using electronic media;
Conflict Resolution: resolving differences in an appropriate manner;
Whole Group Instruction: teacher-led instruction to introduce new concepts on skill building.

Assessment & Evaluation of Student Achievement

Students are provided with opportunities to demonstrate the highest level of their achievement of the expectations in the four achievement categories. The weighting of the categories should comply with the board or school plan.

The weighting applies to evaluations conducted throughout the course (70%) and the final evaluation (30%). The assessment and evaluation for Unit 6 is intended to be counted as final evaluation in addition to a final exam. Application of knowledge skills is a key component of a Computer and Information Science course. A part of the final unit is a joint student/teacher-designed final programming project that brings together many expectations of the course. Students use the software design life cycle to define the solution, analyse needs, plan a solution, and implement and test the solution. This final project is a chance for students to demonstrate the application of acquired knowledge and skills and to use thinking and inquiry skills in the problem-solving process. Sample final projects are: a movie reservation system, a graphics- or text-based game, a programmed tutorial on a computer or non-computer topic, a simulated banking system, and a program designed for a school function, such as event scheduling or a student council record-keeping system.

Expectations are listed in the first unit where they apply. Some expectations are repeated in subsequent units/activities where appropriate.

Students are assessed and evaluated using the following strategies:

Diagnostic: at the beginning of a term, a unit of study, or whenever information about prior learning is useful.

- unit pre-tests;
- skill inventory.

Formative: during learning, ongoing feedback to students of their strengths, weaknesses, and achievement of the expectations.

- communication through journals;
- self-assessment rubrics;
- checklists for programming problems;
- student/teacher conferencing;
- observation;
- peer assessment rubrics;
- quizzes;
- anecdotal comments with suggestions for improvement.

Summative: at the end of a learning process.

- classroom presentations;
- quizzes, tests, unit tests, final exam;
- assignments and projects evaluated using rubrics;
- culminating challenges.

Accommodations

The following are strategies used in the units, more accommodations are included with specific activities:

- referencing and inclusion of recommendations from students' OSRs, IPRCs, and IEPs;
- providing adaptive hardware devices (e.g., large screen monitors, larger fonts, special keyboards);
- providing appropriate environmental accommodations for students with physical disabilities;
- conferencing with Special Education staff and students to discuss modification and accommodation and to ensure physical aspects of the environment meet the needs of students and the program;
- providing word lists, glossaries, definition of terms, and visuals if available;
- grouping weaker students with stronger students to assist in instructional remediation and to provide further challenge;
- allowing more time to organize and complete assignments;
- providing a choice of assignment formats where possible;
- selecting problems that involve programming topics familiar to students to ensure better understanding of requirements (e.g., a student who plays basketball writes a program that keeps basketball statistics);
- providing additional materials to reinforce or extend learning;
- providing opportunities for students requiring enhancement of program;
- using visual and audio-visual aids;
- adjusting expectations for written work and number of assignments required;
- providing for alternative displays of achievement (e.g., oral testing, taped answers, and scribing for students with writing difficulties);
- providing clarification to students of assessment/evaluation tools such as rubrics and checklists;
- selecting groups of varied or similar abilities and skills as appropriate to the activity;
- providing of advanced tutorials and challenges for students with programming experience.

Resources

The following are resources used in many activities; other resources are included with specific activities.

Note: The URLs for the websites have been verified by the writer prior to publication. Given the frequency with which these designations change, teachers should always verify the websites prior to assigning them for student use.

Hume, J.N.P. *Problem Solving and Programming in Turbo Pascal*. Toronto: Holt Software Associates Inc., 1994. ISBN 0-921598-19-X

Hume, J.N.P. *Problem Solving and Programming in Turing*. Toronto: Holt Software Associates Inc., 1993. ISBN 0-921598-16-5

Wright, Peter. *Peter Wright's Beginning Visual Basic 6.0*. Birmingham, UK: Wrox Press. 1998. ISBN 1-861001-05-3

Carter, John. *Problem Solving in Pascal*. Toronto: Addison-Wesley Publishers Limited, 1989, pp. 343, 350. ISBN 0-201-11215-9

Turing programming language information and resources

<http://www.holtsoft.com>

<http://rs6000.georgianc.on.ca/~rodrigo/turing/>

Visual Basic language information and resources

<http://www.dcs.napier.ac.uk/hci/VB50/home.html>

<http://www.vbexplorer.com/>

Pascal programming language information and resources

<http://www8.silversand.net/techdoc/pascal/paslist.htm>

Qbasic programming language information and resources

<http://www.astentech.com/tutorials/QBasic.html>

Website Development Process

<http://www.stratfordinternet.com/process.htm>

Careers and Career Planning

Human Resources Development Canada – <http://www.hrdc-drhc.gc.ca/common/home.shtml>

Career Planning from Yahoo! Canada –

http://ca.yahoo.com/Regional/Countries/Canada/Education/Career_and_Vocational/Career_Planning/

Monster.ca – <http://www.monster.ca>

Workopolis (Globe and Mail Careers) – <http://globecareers.workopolis.com/>

Government of Ontario Training and Jobs website – <http://www.edu.gov.on.ca/eng/training/training.html>

Sympatico.ca’s Careers page – <http://www1.sympatico.ca/Contents/Careers/>

htc Canada’s HiTech Career Journal – <http://www.kaplancareers.com/htc/>

Computer Technology News

Wired News – <http://www.wired.com>

ZDNet – <http://www.zdnet.com>

TechWeb – <http://www.techweb.com>

Canada Computes – <http://www.canadacomputes.com>

cnet – <http://news.cnet.com>

Postsecondary Education

Canadian Universities and Colleges from Yahoo! Canada –

http://ca.yahoo.com/Regional/Countries/Canada/Education/Higher_Education/Colleges_and_Universities

Government of Ontario Post-secondary website –

<http://www.edu.gov.on.ca/eng/general/postsec/postsec.html>

OSS Considerations

The Grade 11 Computer and Information Science course may be used as a compulsory credit (“1 additional credit in science [Grade 11 or Grade 12] or technological education credit [Grades 9-12]” OSS, 1999, p. 9) or as an optional credit. It provides students with an educational base for studies in Grade 12 and postsecondary destinations.

The curriculum emphasizes theory and concrete applications. Teaching/learning strategies and accommodations are selected to meet the needs of all students. Anti-discrimination education, accommodations for exceptional students, career goals/cooperative education, , and community partnerships are addressed in the course. These inclusions support the policies in *Ontario Secondary Schools, Grades 9 to 12: Program and Diploma Requirements, 1999*. Career exploration throughout all units is available in reference to *Choices Into Action: Guidance and Career Education Program Policy for Elementary and Secondary Schools, 1999*.

Coded Expectations, Computer and Information Science, Grade 11, University/College Preparation, ICS3M

Theory and Foundation

Overall Expectations

- TFV.01 · describe at least two problem-solving models;
- TFV.02 · identify the stages in the software development process (problem definition, analysis, design, implementation, testing, and maintenance);
- TFV.03 · explain standard control and data structures used in computer programs;
- TFV.04 · identify on-line and off-line resources;
- TFV.05 · explain the functions of basic computer components.

Specific Expectations

Problem Solving, Logic, and Design

- TF1.01 – define problems by identifying the expected output and necessary user input;
- TF1.02 – evaluate the usefulness of available software tools in a problem-solving situation, using criteria such as ease of use and time required for processing;
- TF1.03 – describe the steps in the software development process and their importance in the development of large programs or applications;
- TF1.04 – explain different problem-solving models (e.g., top-down, bottom-up) that can be used to create a computer program;
- TF1.05 – determine the level of error checking required for given problems;
- TF1.06 – identify the possibilities and limitations of proposed designs;
- TF1.07 – document for the user the potential and limitations of programs.

Programming Concepts

- TF2.01 – describe the characteristics of integer, real, character, and Boolean data types;
- TF2.02 – define constants, variables, expressions, and assignment statements;
- TF2.03 – describe the order in which arithmetic and logical operations are performed;
- TF2.04 – explain the use of Boolean operators in compound expressions;
- TF2.05 – define the structure of one- and two-dimensional arrays and associated concepts (e.g., subscripts, elements, bounds);
- TF2.06 – explain the purpose of selection and repetition structures, and how they are expressed in a programming language;
- TF2.07 – describe the purpose of functions and procedures, and how they are expressed in a programming language;
- TF2.08 – describe parameter passing and scope;
- TF2.09 – identify differences between local and global variables;
- TF2.10 – identify differences among logic, runtime, and syntax errors;
- TF2.11 – describe the evolution of programming languages (e.g., machine, assembly, high-level, 4GL);
- TF2.12 – evaluate available on-line resources such as “readme” files, help files, and “frequently asked questions” files;
- TF2.13 – evaluate available off-line resources such as user manuals and reference manuals;
- TF2.14 – explain the importance of external and internal documentation and programming style;
- TF2.15 – identify common acronyms used in the computing industry.

Hardware, Interfaces, and Networking Systems

TF3.01 – describe the function and location of the basic components of a computer (e.g., motherboard, CPU, I/O devices, memory);

TF3.02 – identify common computer peripheral devices (e.g., mouse, keyboard, screen, printer, multimedia devices) and their primary functions;

TF3.03 – explain differences among software for systems, applications, and programming.

Skills and Processes

Overall Expectations

SPV.01 · develop effective programs by following the steps in the software design process;

SPV.02 · use defined programming practices (e.g., headers, indentation, internal documentation, informative variable names);

SPV.03 · produce appropriate internal and external documentation;

SPV.04 · properly use an operating system, including a network.

Specific Expectations

Problem Solving, Logic, and Design

SP1.01 – resolve ambiguities and missing information in problem statements;

SP1.02 – use the input, process, and output model to solve problems;

SP1.03 – select suitable data structures to represent information;

SP1.04 – develop and maintain a project plan that covers all aspects of the development process for a computer program;

SP1.05 – develop appropriate algorithms in text or diagram form to solve problems and verify solutions;

SP1.06 – produce user-friendly input and output forms;

SP1.07 – solve the same problem using various tools (e.g., a calculator and a computer program, a sort program and a spreadsheet/database/word processor sort function);

SP1.08 – verify solutions to problems;

SP1.09 – incorporate modularity into the design process.

Programming Practices

SP2.01 – use constants, variables, expressions, and assignment statements to store and manipulate numeric, character, and logical data in programs;

SP2.02 – incorporate one-dimensional and two-dimensional arrays into computer programs;

SP2.03 – write programs that use related arrays to store and extract data;

SP2.04 – use selection structures, counted and conditional loops, and nested selection and loop structures;

SP2.05 – manipulate numbers and text using built-in subroutines;

SP2.06 – write subroutines that pass parameters and use local and global variables;

SP2.07 – implement a program design using sequence, selection, and repetition structures;

SP2.08 – use on-line and off-line reference materials effectively;

SP2.09 – adhere to defined programming style, including naming conventions for variables and subroutines, indentation, and spacing;

SP2.10 – incorporate and maintain internal documentation to a specific set of standards, including author, date, file name, purpose, and explanatory comments of major statement groups;

SP2.11 – develop external documentation (including pseudocode, diagrams, and charts) to summarize the design;

SP2.12 – test completed programs with a full range of valid data to ensure that all components work as expected;

-
- SP2.13 – interpret errors during testing and program execution;
 - SP2.14 – trace program execution using manual methods and software debugging tools;
 - SP2.15 – identify and correct logic, runtime, and syntax errors in programs;
 - SP2.16 – use linear searches and simple sort routines in programs;
 - SP2.17 – write programs that access sequential files;
 - SP2.18 – perform peer evaluations of internal documentation and programming style.

Hardware, Interfaces, and Networking Systems

- SP3.01 – use an operating system to perform tasks such as managing files and configuring hardware;
- SP3.02 – use built-in networking functions such as shared files and input/output devices;
- SP3.03 – use common Internetworking services to access and navigate global information resources;
- SP3.04 – develop computer resources to share information globally or locally;
- SP3.05 – implement a comprehensive backup strategy for files.

Impact and Consequences

Overall Expectations

- ICV.01 · explain issues related to the ethical use of computers;
- ICV.02 · describe emergent technologies and their impact on society;
- ICV.03 · identify information systems and computer science career paths, and their educational requirements.

Specific Expectations

Effects of Information Technology

- IC1.01 – explain how the pervasiveness of computer technology affects daily life;
- IC1.02 – describe how information is gathered using computers and how this can affect peoples’ privacy and right to information;
- IC1.03 – identify a number of available sources of career and educational information using networks and evaluate their reliability and accuracy;
- IC1.04 – describe, using presentation software, emergent technologies and their potential influence on society;
- IC1.05 – use appropriate strategies to avoid potential health and safety problems associated with computer use, such as musculo-skeletal disorders and eye strain;
- IC1.06 – explain the importance to identifying career paths of keeping up to date on current articles and thought on computer technology.

Postsecondary Education and Career Opportunities

- IC2.01 – identify postsecondary educational opportunities leading to careers in information systems and computer science, and report on their entry requirements;
- IC2.02 – identify which careers require computer expertise, using local or national media;
- IC2.03 – identify opportunities for apprenticeship training and co-op programs.

Ontario Catholic School Graduate Expectations

The graduate is expected to be:

A Discerning Believer Formed in the Catholic Faith Community who

- CGE1a** -illustrates a basic understanding of the **saving story** of our Christian faith;
- CGE1b** -participates in the **sacramental life** of the church and demonstrates an understanding of the centrality of the Eucharist to our Catholic story;
- CGE1c** -actively reflects on **God’s Word** as communicated through the Hebrew and Christian scriptures;
- CGE1d** -develops attitudes and values founded on Catholic **social teaching** and acts to promote social responsibility, human solidarity and the common good;
- CGE1e** -speaks the **language of life**... “recognizing that life is an unearned gift and that a person entrusted with life does not own it but that one is called to protect and cherish it.” (Witnesses to Faith)
- CGE1f** -seeks intimacy with God and celebrates **communion** with God, others and creation through prayer and worship;
- CGE1g** -understands that one’s purpose or **call in life** comes from God and strives to discern and live out this call throughout life’s journey;
- CGE1h** -respects the **faith traditions**, world religions and the life-journeys of **all people of good will**;
- CGE1i** -integrates faith with life;
- CGE1j** -recognizes that “sin, human weakness, conflict and forgiveness are part of the human journey” and that the cross, the ultimate sign of forgiveness is at the heart of **redemption**. (Witnesses to Faith)

An Effective Communicator who

- CGE2a** -listens actively and critically to understand and learn in light of gospel values;
- CGE2b** -reads, understands and uses written materials effectively;
- CGE2c** -presents information and ideas clearly and honestly and with sensitivity to others;
- CGE2d** -writes and speaks fluently one or both of Canada’s official languages;
- CGE2e** -uses and integrates the Catholic faith tradition, in the critical analysis of the arts, media, technology and information systems to enhance the quality of life.

A Reflective and Creative Thinker who

- CGE3a** -recognizes there is more grace in our world than sin and that hope is essential in facing all challenges;
- CGE3b** -creates, adapts, evaluates new ideas in light of the common good;
- CGE3c** -thinks reflectively and creatively to evaluate situations and solve problems;
- CGE3d** -makes decisions in light of gospel values with an informed moral conscience;
- CGE3e** -adopts a holistic approach to life by integrating learning from various subject areas and experience;
- CGE3f** -examines, evaluates and applies knowledge of interdependent systems (physical, political, ethical, socio-economic and ecological) for the development of a just and compassionate society.

A Self-Directed, Responsible, Life Long Learner who

- CGE4a** -demonstrates a confident and positive sense of self and respect for the dignity and welfare of others;
- CGE4b** -demonstrates flexibility and adaptability;
- CGE4c** -takes initiative and demonstrates Christian leadership;
- CGE4d** -responds to, manages and constructively influences change in a discerning manner;
- CGE4e** -sets appropriate goals and priorities in school, work and personal life;
- CGE4f** -applies effective communication, decision-making, problem-solving, time and resource management skills;
- CGE4g** -examines and reflects on one's personal values, abilities and aspirations influencing life's choices and opportunities;
- CGE4h** -participates in leisure and fitness activities for a balanced and healthy lifestyle.

A Collaborative Contributor who

- CGE5a** -works effectively as an interdependent team member;
- CGE5b** -thinks critically about the meaning and purpose of work;
- CGE5c** -develops one's God-given potential and makes a meaningful contribution to society;
- CGE5d** -finds meaning, dignity, fulfillment and vocation in work which contributes to the common good;
- CGE5e** -respects the rights, responsibilities and contributions of self and others;
- CGE5f** -exercises Christian leadership in the achievement of individual and group goals;
- CGE5g** -achieves excellence, originality, and integrity in one's own work and supports these qualities in the work of others;
- CGE5h** -applies skills for employability, self-employment and entrepreneurship relative to Christian vocation.

A Caring Family Member who

- CGE6a** -relates to family members in a loving, compassionate and respectful manner;
- CGE6b** -recognizes human intimacy and sexuality as God given gifts, to be used as the creator intended;
- CGE6c** -values and honours the important role of the family in society;
- CGE6d** -values and nurtures opportunities for family prayer;
- CGE6e** -ministers to the family, school, parish, and wider community through service.

A Responsible Citizen who

- CGE7a** -acts morally and legally as a person formed in Catholic traditions;
- CGE7b** -accepts accountability for one's own actions;
- CGE7c** -seeks and grants forgiveness;
- CGE7d** -promotes the sacredness of life;
- CGE7e** -witnesses Catholic social teaching by promoting equality, democracy, and solidarity for a just, peaceful and compassionate society;
- CGE7f** -respects and affirms the diversity and interdependence of the world's peoples and cultures;
- CGE7g** -respects and understands the history, cultural heritage and pluralism of today's contemporary society;
- CGE7h** -exercises the rights and responsibilities of Canadian citizenship;
- CGE7i** -respects the environment and uses resources wisely;
- CGE7j** -contributes to the common good.

Unit 3: Problem Solving with Procedures and Functions

Time: 18 hours

Unit Description

This unit focuses on program modularity and career exploration. Students write programs using existing sub-programs and then progress to writing programs including their own sub-programs. They also explore careers in computer studies and develop skills in program modularity (e.g., a program to encrypt/decrypt a passage of text using substitution skills encoding). Students complete a reflection on work and on the *Laborem Exercens* encyclical.

Unit Synopsis Chart

Activity	Time	Expectations	Assessment	Tasks
1. The Software Development Process	1 hour	TFV.02, TF1.03 CGE7a	K	In-class discussion Homework
2. Investigating Math and Text Subroutines	3 hours	TF2.07, SP2.05 CGE2b	K; I	Lab exercises Quiz
3. Defining Our Own Subroutines	2 hours	SP2.06, SP1.09, TF2.08, TF2.09	K; I; A	Lab exercises Homework Quiz
4. Exploring Careers in Computing	3 hours	ICV.03, IC1.06, IC2.01, IC2.02, IC2.03; CGE5b	K; I	Research Presentation
5. Programming with Subroutines	7 hours	SPV.01, SP2.06 CGE2b, 5a	I; C; A	Assignment Test
6. Reflecting on Work	2 hours	CGE1g, CGE2b, CGE4g, CGE5b, CGE5d	K;I; C; A	Research In-class discussion Reflection paper

Activity 1: The Software Development Process

Time: 60 minutes

Description

Students are introduced to the software development process. They learn the different stages of the process and the important considerations at each stage. Students compare the software development process with other development processes in other industries and other educational areas. They look at how development processes are applied in large-scale projects.

Strand(s) & Learning Expectations

Ontario Catholic School Graduate Expectations

CGE7a - acts morally and legally as a person formed in Catholic tradition.

Strand(s): Theory and Foundation

Overall Expectations

TFV.02 - identify the stages in the software development process (problem definition, analysis, design, implementation, testing, and maintenance).

Specific Expectations

TF1.03 - describe the steps in the software development process and their importance in the development of large programs or applications.

Prior Knowledge & Skills

Students:

- have written several programs;
- have used several large and complex application programs (such as a word processor or spreadsheet).

Planning Notes

- Select from the following possible preparations for this activity:
 - Gather examples (possibly from videos, articles, or web sites) of development processes in other industries (e.g., auto industry and food industry).
 - Consult with broad-base technology teachers about development processes used in their courses and relevant industries.
 - Inventory the programs or applications used by students and staff in the school.
 - Gather examples of the software development process in the creation/production of large programs.
 - Contact a computer professional (programmer, analyst, or project manager) at a local company and inquire about the use of the software development process within the company.
- Prepare a handout, overhead presentation, multimedia presentation, or web page on the software development process.
- Prepare in-class task to help students understand different steps in the software development process.
- Prepare a homework assignment to help students understand the software development process.

Teaching/Learning Strategies

- Introduce the concept of a development process and present examples of how products are created.
- Introduce the concept of the software development process and stages of this process.
- Present a development process from another industry and compare it to the software development process.
- Discuss how needs of various industries shape their design processes.
- Students participate in a class discussion about the development process.
- Facilitate discussion on students' programming experience and how they have carried out programming tasks.
- Query students about the programs they have used at school, at home, or at work.
- Facilitate discussion regarding the creation of large programs or applications (e.g., word processor, spreadsheet, photo-editing tools, web browser, and operating system).
- Present examples of the use of the software development process in the software industry.
- Discuss the ethical implications of the design process.

Assessment & Evaluation of Student Achievement

The teacher and students gather assessment information based on specific expectations outlined for this activity, including:

- a formative assessment of the assigned work in the form of roving conferences as students work on in-class task (a checklist, based on Appendix 3.1.1, could be used).

Accommodations

The following are ways in which the activity can be adapted to accommodate exceptional students' needs:

- outline on overhead and/or print a list of programs used and the software development process to assist in focus on the task and discussion.

Resources

Website Development Process – <http://www.stratfordinternet.com/process.htm>

Product Development Process – <http://www.rt66.com/~korteng/plan.htm>

Angotti Product Development Page – <http://www.angotti.com/#4>

Business Technology Associates, Software Development Cycle – <http://www.biztechwiz.com/btasdlc.html>

The Product Development Cycle – <http://www.xed.co.uk/design.htm>

Activity 2: Investigating Math and Text Subroutines

Time: 180 minutes

Description

Students explore pre-defined mathematical and text subroutines made available by their programming software. Through a series of lab exercises, students practise using these pre-defined subroutines and reflect on how they could be used to help solve a larger programming problem.

Strand(s) & Learning Expectations

Ontario Catholic School Graduate Expectations

CGE2b - reads, understands, and uses written materials effectively.

Strand(s): Theory and Foundation, Skills and Processes

Specific Expectations

TF2.07 - describe the purpose of functions and procedures, and how they are expressed in a programming language;

SP2.05 - manipulate numbers and text using built-in subroutines.

Prior Knowledge & Skills

Students:

- are aware of mathematical functions (round, sqrt, sin, cos, tan, etc.);
- are aware of spreadsheet applications;
- write programs that accept input from users, process data using formulas, and display results;
- are able to debug programs containing syntax and logic errors;
- identify string as a variable data type;
- recognize the ASCII table of values representing keyboard characters.

Planning Notes

- Make connections to problem-solving strategies previously introduced.
- Provide on-line and/or print resources for shared use by students.
- Gather material about mathematical functions and built-in functions in items such as calculators, spreadsheets, and word processors for comparison.

Teaching/Learning Strategies

- Introduce/review the “divide and conquer” problem-solving strategy; brainstorm large problems currently facing students (e.g., ISU completion) and how this task could be divided into smaller, more manageable sub-problems.
- Facilitate discussion on how this strategy can be applied to programming (e.g., how it may help to tackle the daunting task of creating a major programming solution such as a spreadsheet application).
- Present examples (in the form of handouts, overheads, or web-based presentation) of pre-defined subroutines provided in the programming language used.
- Emphasize the important task of the programmer to research built-in tools available to them.
- Students use on-line help and/or print resources to find and investigate usefulness of pre-defined mathematical subroutines, e.g., a pre-defined subroutine written to perform any/all of these tasks:
 - a) round any value to a specified number of decimal places;
 - b) find the square root of a value;
 - c) generate a random integer between any lower/upper parameters specified;
 - d) calculate payment for a bank loan (given the interest rate, term of loan, and principal amount).
 - e) count the length of a string;
 - f) produce a string of repeating characters (e.g., ten asterisks);
 - g) change the case of characters in a string (lowercase/uppercase/proper);
 - h) find the position of a specified character within a string;
 - i) return a substring of a specified length from the beginning, end, or specified starting position of a longer string;
 - j) return the position of a specified substring from within a longer string.
- Demonstrate built-in subroutines of type **function** and subroutines of type **procedure**; ask students to identify differences between the two types of subroutines,
e.g., $\text{sqrt}(\text{variable})$ vs. $\text{Circle}(x, y, \text{radius})$
 $\text{length}(\text{variable})$ $\text{Line}(x, y)$
- Solicit responses from students and verify that functions return a value vs. procedures, which perform a series of steps but do not return a value. Demonstrate the use of variable tracking or desk tracing.
- Students record new tools and strategies in a journal or notes.
- Students complete as many lab exercises as possible in time allotted. (See Appendix 3.2.1.)

Assessment & Evaluation of Student Achievement

The teacher and students gather assessment information based on specific expectations outlined for this activity, including:

- a formative assessment of the assigned work in the form of roving conferences;
- a peer and self-evaluation of the students’ work, using a checklist of criteria created by the teacher and/or students (Appendix 3.2.2);
- a summative assessment in the form of a quiz.

Accommodations

The following are ways in which the activity can be adapted to accommodate exceptional students’ needs:

- provide print copies of subroutines and specific vocabulary new to students;
- draw/chart investigation findings to assist students in multi-faceted task;
- selectively pair/group students to assist with recording results (i.e., classifying and investigating);

-
- provide “scaffolded” programs (i.e., program listings with subroutine headings, loop structures, etc. already included) to help struggling students;
 - use peer/mentor support for recording results and journal entries.

Resources

Installed help files and/or manuals for the programming language in use.

Turing programming language information and resources

<http://www.holtsoft.com>

<http://rs6000.georgianc.on.ca/~rodrigo/turing/>

Visual Basic language information and resources

<http://www.dcs.napier.ac.uk/hci/VB50/home.html>

<http://www.vbexplorer.com/>

Pascal programming language information and resources

<http://www8.silversand.net/techdoc/pascal/paslist.htm>

Qbasic programming language information and resources

<http://www.astentech.com/tutorials/QBasic.html>

Activity 3: Defining Our Own Subroutines

Time: 120 minutes

Description

This activity builds on students’ knowledge of variables and subroutines. Through a series of discussions, demonstrations, and lab exercises, students learn to make the distinction between local and global variables as they relate to variable parameters in programmer-defined subroutines.

Strand(s) and Learning Expectations

Strand(s): Theory and Foundation

Specific Expectations

TF2.07 - describe the purpose of functions and procedures, and how they are expressed in a programming language;

TF2.08 - describe parameter passing and scope;

TF2.09 - identify differences between local and global variables.

Planning Notes

- Gather samples of built-in subroutines for identification of functions vs. procedures and for identification of parameters.
- Provide and demonstrate syntax required for writing programmer-defined subroutines in the language of choice.
- Provide on-line and/or print resources for shared use by students.
- Provide examples that help students understand variable scope and the passing of parameters as information to parts of the program. Students often have difficulty with variable scope and parameter passing.
- Review variable naming conventions and/or adopt conventions that identify the scope of a variable.

Prior Knowledge & Skills

Students:

- can demonstrate appropriate use of built-in subroutines;
- are able to debug programs containing syntax and logic errors.

Teaching/Learning Strategies

- Discuss with students the concept that a subroutine (function or procedure) often needs data in order to complete its task. Data is given (passed) to a procedure by enclosing it in parentheses in the procedure **call**.
e.g., in order for the sqrt function to produce the desired result, a number (**parameter**) must first be **passed** to the sqrt subroutine: sqrt(36)
and a value is **returned**: 6
- Assign students the task of identifying the required parameters for a series of selected functions and procedures, making connections to math and text subroutines explored in Activities 2 and 3.
- Discuss the concept of variable scope and the need for parameters to pass information from one component to another.
- Facilitate a brainstorming session to write an algorithm for a modular program that reads (input) a customer's bank *balance* then, if the balance is less than zero, prints an "Overdraft" notice, or else *interest* is calculated (based on current *rate*) and added to the original balance.
- Promote use of subroutines to divide tasks, identifying required parameters for each subroutine.
- Introduce the concept that the parameters for *balance* and *interest* are considered **variable** parameters because their values are changed by the Calculate subroutine, while the *rate* parameter is not a variable parameter because its value is not changed by the Calculate subroutine.
- Demonstrate the syntax for defining a custom subroutine in your programming language.
- Student record new tools and strategies in a journal or notes.
- Student complete as many lab exercises as possible in time allotted. (See Appendices 3.3.1, 3.3.2, and 3.3.3)

Assessment & Evaluation of Student Achievement

The teacher and students gather assessment information based on specific expectations outlined for this activity, including:

- a formative assessment of the assigned work in the form of roving conferences;
- a formative assessment of the journal/notebook (checking for inclusion of tools/strategies presented);
- a summative assessment of the student's ability to describe parameter passing and scope and identify differences between local and global variables. Students break down computer tasks into specific functions and define the modules, parameters, and variables used.

Accommodations

The following are ways in which the activity can be adapted to accommodate the exceptional students' needs:

- support brainstorming activity through use of web or mapping chart with print copy;
- provide support through one-to-one teacher-directed conferencing to ensure understanding;
- provide "scaffolded" programs (program listings with subroutine headings, loop structures, etc., already included) to help struggling students.

Resources

Installed help files and/or manuals for the programming language in use.

See Resources in Activity 2.

Activity 4: Exploring Careers in Computing

Time: 180 minutes

Description

Students investigate career paths leading to information systems and computer science careers. They research the educational requirements of a chosen career and look into the availability of programs in colleges, universities, and private institutions that will help them meet those requirements. Students also investigate apprenticeship and co-op programs.

Strand(s) & Learning Expectations

Ontario Catholic School Graduate Expectations

CGE5b - thinks critically about the meaning and purpose of work.

Strand(s): Impact and Consequences

Overall Expectations

ICV.03 - identify information systems and computer science career paths, and their educational requirements.

Specific Expectations

IC1.06 - explain the importance to identifying career paths of keeping up to date on current articles and thought on computer technology;

IC2.01 - identify postsecondary educational opportunities leading to careers in information systems and computer science, and report on their entry requirements;

IC2.02 - identify which careers require computer expertise, using local or national media;

IC2.03 - identify opportunities for apprenticeship training and co-op programs.

Planning Notes

- Obtain a list of different computer-related careers.
- Gather articles from newspapers, magazines, and the Web that relate to computer technology and computer-related careers (with some emphasis on trends).
- Research the different programs available in community colleges, universities, and private institutions (with emphasis on the entry requirements to these programs).
- Research apprenticeship training opportunities and co-op programs for computer-related careers.
- Gather material, with assistance from the Guidance/Student Services Department, regarding different career paths leading to careers in information systems and computer science.
- Contact the Guidance/Student Services Department to make a presentation to the class regarding computer-related programs in community colleges, universities, and private institutions.
- Contact the Co-op Department to make a presentation to the class regarding apprenticeship opportunities and co-op programs.

Prior Knowledge & Skills

Students:

- are aware of some computer-related careers;
- are able to use a web browser;
- are aware of the services provided by the Guidance/Student Services Department;
- are aware of the services provided by the Co-op Department.

Teaching/Learning Strategies

- Present material (from books, newspapers, magazines, web pages, or short video) on computer-related careers.
- Students investigate current trends in computer-related careers through the use of print material (books, newspaper, magazines) and/or online resources.
- Facilitate discussion of the ethical considerations of computer careers (e.g., how they may be asked to develop technology that effects privacy or copyright).
- Facilitate discussion on the importance of having computer skills (e.g., ask students about the skills that their parents are required to have at their workplace).
- The teacher discusses various strategies for keeping up-to-date on current computer technology.
- Introduce computer-related programs in community colleges, universities, and private institutions (possibly with the help of the Guidance/Student Services representative).
- Students compare the entry requirements to computer-related programs in community colleges, universities, and private institutions.
- Discuss computer-related apprenticeship opportunities and co-op programs (possibly with the help of the Co-op Department).
- Students distinguish between and compare apprenticeship and co-op programs.
- Students complete an assessment task (possibly one from the list in Assessment & Evaluation of Student Achievement) demonstrating their knowledge of the material presented in this section. (See Appendix 3.4.1.)

Assessment & Evaluation of Student Achievement

The teacher and students gather assessment information based on specific expectations outlined for this activity including:

- a summative assessment of the assigned work in the form of one the following:
 - bulletin board project that shows trends in computer-related careers;
 - a radio advertisement from a fictional company seeking candidates for a computer-related job;
 - a print advertisement on a possible future computer-related career;
 - a private detective's report regarding an investigation into entry requirements to computer-related programs at a chosen college, university, or private institution;
 - a web page that has links to the different colleges, universities, and private institutions and their computer-related programs.

See Appendix 3.4.2.

Accommodations

The following are ways in which the activity can be adapted to accommodate the student's needs:

- use mapping and/or flowcharts to assist students in recording information;
- conference with different individuals/groups to ensure "on-task" focus re: data collection;
- provide outline in print for entry of information on comparisons (i.e., various school programs);
- group students who are new to the country with other students so they are not held back by language or lack of cultural experiences.

Resources

Careers and Career Planning

Human Resources Development Canada – <http://www.hrdc-drhc.gc.ca/common/home.shtml>

Career Planning from Yahoo! Canada –

http://ca.yahoo.com/Regional/Countries/Canada/Education/Career_and_Vocational/Career_Planning/

Monster.ca – <http://www.monster.ca>

Workopolis (Globe and Mail Careers) – <http://globecareers.workopolis.com/>
Government of Ontario Training and Jobs website – <http://www.edu.gov.on.ca/eng/training/training.html>
Sympatico.ca’s Careers page – <http://www1.sympatico.ca/Contents/Careers/>
htc Canada’s HiTech Career Journal – <http://www.kaplancareers.com/htc/>

Computer Technology News

Wired News – <http://www.wired.com>
ZDNet – <http://www.zdnet.com>
TechWeb – <http://www.techweb.com>
Canada Computes – <http://www.canadacomputes.com>
cnet – <http://news.cnet.com>

Postsecondary education

Canadian Universities and Colleges from Yahoo! Canada –
http://ca.yahoo.com/Regional/Countries/Canada/Education/Higher_Education/Colleges_and_Universities
Government of Ontario Post-secondary website –
<http://www.edu.gov.on.ca/eng/general/postsec/postsec.html>
Local Chamber of Commerce

Activity 5: Programming with Subroutines

Time: 420 minutes

Description

Students demonstrate an understanding of predefined and user-defined subroutines. They work in small groups to accomplish a programming task that requires the use of several user-defined subroutines. Students document their use of the software development process to solve their problem.

Strand(s) & Learning Expectations

Ontario Catholic School Graduate Expectations

CGE2b - reads, understands, and uses written materials effectively;
CGE5a - works effectively as an interdependent team member.

Strand(s): Skills and Processes

Overall Expectations

SPV.01 - develop effective programs by following the steps in the software design process.

Specific Expectations

SP2.06 - write subroutines that pass parameters and use local and global variables.

Planning Notes

- Prepare a programming problem for students to solve. This problem must have a solution that requires the definition of several subroutines. (See Appendix 3.5.1 for a sample problem.)
- Prepare a test to evaluate students’ understanding of the concepts presented during this unit.
- Review Programming with Subroutine notes (Appendix 3.5.2).

Prior Knowledge & Skills

Students:

- have used predefined subroutines;
- have created several subroutines.

Teaching/Learning Strategies

- Review the software development process.
- Divide the class into groups of two or three students.
- Remind students of the importance of respect for all group members and responsibility of all group members.
- Facilitate discussion about the programming problem (problem definition stage).
- Lead students in analysing the problem (analysis stage).
- Students participate in small-group discussion to design possible solutions to the programming problem (design stage).
- Students examine and assess each of the possible solutions (design stage).
- Students select the best solution (design stage).
- Students describe and define the subroutines required to solve the problem (design stage).
- Students participate in a group decision-making process to assign the creation of subroutines to different group members (implementation stage).
- Students create, program, and document a subroutine (implementation stage).
- Students test the subroutine (testing stage).
- Students combine their subroutine with the other group members' subroutines to create a solution to the problem (implementation stage).
- Students participate in the documentation of the solution (implementation stage).
- Assess the students' progress regularly as they create a solution to the problem.
- Students test the group's solution (testing stage).
- Students assess and analyse the group's solution (maintenance stage).
- Administer a test to assess students' understanding of concepts presented in this unit.

Assessment & Evaluation of Student Achievement

The teacher and students gather assessment information based on specific expectations outlined for this activity, including:

- a summative assessment of the assigned work in the form of a group assignment to combine and consolidate the concepts presented during this unit (see Appendix 3.5.3 for an example of a rubric that might be used);
- a summative assessment of the assigned work in the form of a test to assess students' understanding of the concepts presented during this unit.

Accommodations

The following are ways in which the activity can be adapted to accommodate the student's needs:

- select groups to allow for both remediation and enrichment;
- encourage sharing of individual data;
- provide parts of subroutines or parts of code to aid struggling groups (see Appendix 3.5.2).

Resources

Installed help files and/or manuals for the programming language in use.

See Resources in Activity 2.

Activity 6: Reflecting on Work

Time: 120 minutes

Description

Students think critically about the meaning and purpose of work. They explore the Catholic Church's view on work and its importance to human life or those held by other groups in our society. Students examine and reflect on their personal attitude towards work. They reflect on the relationship between technology and work and how both can be used for the greater good.

Strand(s) & Learning Expectations

Ontario Catholic School Graduate Expectations

CGE1g - understands that one's purpose or call in life comes from God and strives to discern and live out this call throughout life's journey;

CGE2b - reads, understands, and uses written materials effectively;

CGE4g - examines and reflects on one's personal values, abilities, and aspirations influencing life's choices and opportunities;

CGE5b - thinks critically about the meaning and purpose of work;

CGE5d - finds meaning, dignity, fulfillment, and vocation in work which contributes to the common good.

Planning Notes

- Read the Catholic Church's encyclical on the dignity of work – *Laborem Exercens* (see Appendix 3.6.1). This is available in print and on various websites.
- Reflect on what is stated in the encyclical.
- Select parts of the encyclical to present to students (see Appendix 3.6.1).
- Prepare a handout that can be used during an in-class discussion (see Appendix 3.6.2).
- Prepare an assignment to help students further reflect on the contents of the encyclical (see Appendix 3.6.2).

Prior Knowledge & Skills

Students:

- are aware of the increasing use of computer technology in the workplace;
- are aware that computer technology may displace workers.

Teaching/Learning Strategies

- Facilitate discussion on the meaning of work and students' personal attitudes towards work.
- Facilitate discussion on students' ambitions and aspirations.
- Students participate in a small group discussion regarding the meaning of work.
- Introduce the Catholic Church's view of work and present selected text from the *Laborem Exercens* encyclical.
- Solicit students' reaction to the encyclical.
- Students explore and reflect on the Catholic Church's view on the relationship of technology and work in a written reflection paper (see Appendix 3.6.2).

Assessment & Evaluation of Student Achievement

The teacher and students gather assessment information based on specific expectations outlined for this activity, including:

- a summative assessment of the assigned work in the form of a reflection paper based on the contents of the encyclical presented during this unit or other positions on this issue that have been considered.

Accommodations

The following are ways in which the activity can be adapted to accommodate exceptional students' needs:

- clarify language through print form and simplify meaning of terms for understanding of Church views;
- allow for sharing of reflection as appropriate to support thoughts and/or emotional response of individual students;
- adjust length and/or scope of reflection paper to enrich or remediate as appropriate.

Resources

Laborem Exercens (full text – from a Vatican website) –

http://www.vatican.va/holy_father/john_paul_ii/encyclicals/documents/hf_jp-ii_enc_14091981_laborem-exercens_en.html

Laborem Exercens (full text – alternate site) - <http://www.osjspm.org/cst/le.htm>

Laborem Exercens (in everyday language) - http://www.osjspm.org/cst/le_el.htm

Catholic Social Teaching: John Paul II, *Laborem Exercens* –

<http://catholiceducation.org/articles/religion/re0286.html>

Appendix 3.1.1

The Software Development Process – Assignment Rubric

Criteria	Level 1 (50 – 59%)	Level 2 (60 – 69%)	Level 3 (70 – 79%)	Level 4 (80 – 100%)
Identify the stages of the development process TFV.02	- demonstrates limited knowledge of the stages in the software development process	- demonstrates some knowledge of the stages in the software development process	- demonstrates considerable knowledge of the stages in the software development process	- demonstrates thorough knowledge of the stages in the software development process
Describe the steps in the development process TF1.03	- demonstrates limited ability to describe the steps in the development process	- demonstrates some ability to describe the steps in the development process	- demonstrates considerable ability to describe the steps in the development process	- demonstrates a high level of ability to describe the steps in the development process
Explain the importance of the steps in the development process in the development of large programs TF1.03	- uses thinking skills with limited effectiveness to explain the importance of the steps in the development process in the development of large programs	- uses thinking skills with moderate effectiveness to explain the importance of the steps in the development process in the development of large programs	- uses thinking skills with considerable effectiveness to explain the importance of the steps in the development process in the development of large programs	- uses thinking skills with a high degree of effectiveness to explain the importance of the steps in the development process in the development of large programs

Note: A student whose achievement is below level 1 (50%) has not met the expectations for this assignment or activity.

Appendix 3.2.1

Investigating Math and Text Subroutines – Sample Lab Exercises

Utilize pre-defined mathematical subroutines available in your programming environment to solve the following problems:

1. Input any real number and output:
 - a) the number rounded to a whole number;
 - b) its square root.
2. Input any two numbers and output the positive difference between them.
3. Create a program that calculates and outputs the trigonometric functions sine, cosine, and tangent of any angle input by the user (expressed in radians). The result is rounded to a whole number.
Note: to convert degrees to radians: $\text{radians} = (\pi / 180) * \text{degrees}$ ($\pi = 3.14$)
Challenge: output results in degrees ($\text{degrees} = \text{radians} * (180 / \pi)$)
4. Create a Payment Calculator program to determine monthly payments for a car loan. Input the current interest rate, term of the loan (expressed in years), and principal amount.
5. Input two words and store each in memory as a separate variable. Determine the length of each of the words and output the longer of the two.
6. Input two words and store each in memory as a separate variable. Determine the ASCII value of the first letter of each word. Output the word that begins with the highest ASCII value first, followed by the word that begins with the lower ASCII value.
E.g., Sample Input: berry
 apple
 Sample Output: apple comes before berry
Test your program again, inputting the following data: **apple** **Berry**
What were the results? Why do you think it is important to make connections to ASCII table values?
7. Input a string. Convert to all CAPS (uppercase) and output.
8. Input a word. Isolate the first letter and output it as a capital. Concatenate (join) with the remainder of the word.
9. Ask the user to input his or her name. Output the letters in the name in reverse order (e.g., Lucy becomes ycuL).
10. Input a sentence. Count the number of spaces in order to determine the number of words in the sentence. Output the result. (Was your count accurate?)
11. Input a word. Output the first vowel found and its position within the string.
E.g., Sample Input: VISUAL
 Sample Output: A(n) i was found at position 2.

Appendix 3.2.2

Subroutine Usage Checklist

- | | | |
|---|-----|----|
| 1. Subroutines are selected for appropriate tasks. | Yes | No |
| 2. Correct information is passed to subroutines. | Yes | No |
| 3. Information is submitted to subroutines in the correct sequence. | Yes | No |
| 4. Correct variables are selected in the calling program. | Yes | No |
| 5. Information from subroutines is used correctly. | Yes | No |

Appendix 3.3.1

Defining Our Own Subroutines – Sample Lab Exercises

Write a programmer-defined subroutine for the following:

1. Function CelsiusToFahrenheit (degrees as real) returns real
return ((degrees * (9 / 5)) + 32)
end Function
2. Function FahrenheitToCelsius (degrees as real) returns real
return ((degrees - 32) * (5 / 9))
endFunction
3. Function MilesToKms (miles as real) returns real
return (miles * 1.6)
end Function
4. Function KilobytesToBits (k as real) returns real
return(k * 1024 * 8)
end Function

Create a bank balance program, based on the algorithm created earlier in this activity, defining programmer-defined subroutines as appropriate.

Appendix 3.3.2

Creating Subroutines – Sample Assignment

A program is required to calculate the average of four mid-term marks for a secondary school student. The average should include decimals; the four marks are integers. Describe a subroutine that could be used to calculate the average. Identify the parameters, subroutine task, and return value. List local and global variables and indicate how they would match up between the subroutine and main program.

Appendix 3.3.3

Creating Subroutines – Sample Rubric

Criteria	Level 1 (50 – 59%)	Level 2 (60 – 69%)	Level 3 (70 – 79%)	Level 4 (80 – 100%)
Describe the purpose of functions and procedures, and how they are expressed in a programming language TF2.07	- demonstrates limited ability to describe the purpose of the functions and procedures	- demonstrates some ability to describe the purpose of the functions and procedures	- demonstrates considerable ability to describe the purpose of the functions and procedures	- demonstrates thorough ability to describe the purpose of the functions and procedures
Describe the purpose of functions and procedures, and how they are expressed in a programming language TF2.07	- demonstrates limited ability to express the subroutine in a programming language	- demonstrates some ability to express the subroutine in a programming language	- demonstrates considerable ability to express the subroutine in a programming language	- demonstrates thorough ability to express the subroutine in a programming language
Describe parameter passing and scope and local and global variables. TF2.08, TF2.09	- demonstrates limited knowledge of use of parameters	- demonstrates some knowledge of use of parameters	- demonstrates considerable knowledge of use of parameters	- demonstrates thorough knowledge of use of parameters

Note: A student whose achievement is below level 1 (50%) has not met the expectations for this assignment or activity.

Appendix 3.4.1

Exploring Careers in Computing – Sample Assignments

Radio Ad I

Imagine that you are a member of an advertisement agency. Your agency has been asked to create a 30-second radio ad for a fictional company. The company is advertising openings in their Information Technology department. Mention some of the following points in your ad:

- the job title associated with the job opening;
- the duties associated with the job opening;
- the minimum educational requirements for the job.

Choose an appropriate name for the company and suitable background music for the ad.

Radio Ad II

Imagine that you are a member of an advertisement agency. Your agency has been asked to create a 30-second radio ad for a college, university, or private institution. The institution is advertising its programs (including apprenticeship and co-op programs). Mention some of the following points in your ad:

- features of the institution's program;
- the program's entry requirements;
- the skills that the graduate will possess at program completion.

Choose suitable background music for the ad.

Private Detective's Report

Imagine that you are a private detective. You have just been chosen to investigate the entry requirements for a computer-related program at a college, university, or private institution. Choose a Canadian or International institution to investigate. Once your research is complete, create a report that is suitable for posting on a bulletin board. The report should mention the following:

- the name of the institution;
- the location of the institution;
- the name of the computer-related program;
- the entry requirements;
- the cost of the program.

Magazine or Newspaper Ad

Imagine that you are a member of an advertisement agency. Your agency has been asked to create a one-page (8.5" x 11") ad for a fictional company. The company is advertising openings in their Information Technology department. Mention some of the following points in your ad:

- the job title associated with the job opening;
- the duties associated with the job opening;
- the minimum educational requirements for the job.

Appendix 3.4.2

Exploring Careers in Computing – Formative Evaluation Rubric

Criteria	Level 1 (50 – 59%)	Level 2 (60 – 60%)	Level 3 (70 – 79%)	Level 4 (80 – 100%)
Identify computer-related careers and their educational requirements ICV.03, IC2.02	- demonstrates limited knowledge of computer-related careers and their educational requirements	- demonstrates some knowledge of computer-related careers and their educational requirements	- demonstrates considerable knowledge of computer-related careers and their educational requirements	- demonstrates thorough knowledge of computer-related careers and their educational requirements
Identify postsecondary educational opportunities and their entry requirements IC2.01	- demonstrates limited knowledge of postsecondary educational opportunities and their entry requirements	- demonstrates some knowledge of postsecondary educational opportunities and their entry requirements	- demonstrates considerable knowledge of postsecondary educational opportunities and their entry requirements	- demonstrates thorough knowledge of postsecondary educational opportunities and their entry requirements
Identify apprenticeship training and co-op programs IC2.03	- demonstrates limited knowledge of apprenticeship training and co-op programs	- demonstrates some knowledge of apprenticeship training and co-op programs	- demonstrates considerable knowledge of apprenticeship training and co-op programs	- demonstrates thorough knowledge of apprenticeship training and co-op programs
Explain the importance of keeping up to date on computer technology IC1.06	- uses thinking skills with limited effectiveness to explain the importance of keeping up-to-date on computer technology	- uses thinking skills with moderate effectiveness to explain the importance of keeping up-to-date on computer technology	- uses thinking skills with considerable effectiveness to explain the importance of keeping up-to-date on computer technology	- uses thinking skills with a high degree of effectiveness to explain the importance of keeping up-to-date on computer technology

Note: A student whose achievement is below level 1 (50%) has not met the expectations for this assignment or activity.

Appendix 3.5.1

Programming with Subroutines – Sample Assignment

Number system conversions

The purpose of this project is to combine and consolidate the programming concepts learned in this unit. The student must demonstrate an understanding of the following concepts:

- proper definition of a subroutine;
- parameters;
- local and global variables;
- use of predefined subroutines;
- string concatenation;
- remainder and integer division operator.

It is assumed that the language the student is using has the following predefined functions.

- substring – to select parts of a string;
- string to integer – to convert a string to an integer;
- integer to string – to convert an integer to a string;
- length – to find the length of a string.

If students are not familiar with number systems, a short introduction to binary (base 2) and hexadecimal (base 16) number systems is necessary. Algorithms for converting between binary, decimal, and hexadecimal number systems should be introduced and provided in a handout or a web page.

Students work in groups of two or three.

Programming Task

The binary and hexadecimal number systems are very important in Computer Science. A computing professional comes across these number systems over and over again in computer programming and networking (e.g., network card addresses are usually expressed as a series of hex digits, IP addresses are binary numbers usually expressed as decimal numbers). The ability to work in these number systems is an essential skill in computing.

Working in teams of two or three students, write a program that allows a user to specify a number (decimal, binary, or hexadecimal) and a base to convert to (decimal, binary, or hexadecimal). A subroutine must be written for each of the conversions (decimal to binary, decimal to hexadecimal, binary to decimal, binary to hexadecimal, hexadecimal to decimal, and hexadecimal to binary). The task of writing the subroutines is divided among the members of the group.

Appendix 3.5.2

Programming with Subroutines – Notes

Binary to Decimal

This function demonstrates the algorithm used for converting a binary number into a decimal number.

Function (BinaryNumber is String) returns Integer

Var DecimalNumber is Integer initialized as 0

Var Power is Integer initialized as 0

Var BinaryDigit is Integer

For (I (start value is length of BinaryNumber) to (end value is 1))

BinaryDigit = StringToInteger (BinaryNumber (I))

- BinaryNumber (I) represents the Ith character in the string – this is similar to the substring function

DecimalNumber = DecimalNumber + (BinaryDigit * 2 ** Power)

*- ** is the “power of” operator*

Power = Power + 1

End for

Return DecimalNumber

End Function

Decimal to Binary

This function demonstrates the algorithm used for converting a decimal number into a binary number.

Function DecimalToBinary (DecimalNumber is Integer) returns String

Var BinaryString is String initialized as “”

Var BinaryDigit is Integer

Var DecimalN is Integer initialized as DecimalNumber

Loop

BinaryDigit = DecimalN mod 2

- mod is the remainder function

- the right-hand side will result in 0 or 1

BinaryNumber = IntegerToString (BinaryDigit) + BinaryNumber

- IntegerToString is a predefined function to convert an integer to a string

- The ‘+’ here is the concatenation operator

Exit when DecimanN is 0

DecimalN = DecimalN div 2

- div is the integer division operator

- we want to divide DecimalN by 2 and “throw” away the fractional part

End loop

Return BinaryNumber

End Function

Additional Resources

Binary/Hexadecimal Tutorial – <http://vwop.port5.com/beginner/bhextut.html>

The Hexadecimal Number System – <http://www.cths.nsw.edu.au/kla/IT/tutorial/hexadecimal.htm>

Appendix 3.5.3

Programming with Subroutines – Summative Evaluation Rubric

Criteria	Level 1 (50 – 59%)	Level 2 (60 – 69%)	Level 3 (70 – 79%)	Level 4 (80 – 100%)
Follow the software design process SPV.01	- demonstrates limited ability to effectively use the software design process to solve a computer programming problem	- demonstrates some ability to effectively use the software design process to solve a computer programming problem	- demonstrates considerable ability to effectively use the software design process to solve a computer programming problem	- demonstrates a high level of ability to effectively use the software design process to solve a computer programming problem
Describe the purpose of subroutines TF2.07	- demonstrates limited ability to describe the purpose of subroutines used	- demonstrates some ability to describe the purpose of subroutines used	- demonstrates considerable ability to describe the purpose of subroutines used	- demonstrates a high level of ability to describe the purpose of subroutines used
Use built-in subroutines SP2.05	- demonstrates limited ability to use built-in subroutines where appropriate	- demonstrates some ability to use built-in subroutines where appropriate	- demonstrates considerable ability to use built-in subroutines where appropriate	- demonstrates a high level of ability to use built-in subroutines where appropriate
Write subroutines that pass parameters and use local and global variables SP2.06	- demonstrates limited ability to write subroutines that pass parameters and use local and global variables	- demonstrates some ability to write subroutines that pass parameters and use local and global variables	- demonstrates considerable ability to write subroutines that pass parameters and use local and global variables	- demonstrates a high level of ability to write subroutines that pass parameters and use local and global variables

Note: A student whose achievement is below level 1 (50%) has not met the expectations for this assignment or activity.

Appendix 3.6.1

Reflecting on Work – Notes on *Laborem Exercens*

The purpose of this activity is to give students an opportunity to reflect on their views about work and present the Church's view. Another focus is to encourage discussion regarding the relationship of technology, work, and workers.

Laborem Exercens is an important document that talks about the Church's social doctrine on human work. It is important for all of us (students and teachers) to understand the Church's position on the dignity of human work. The encyclical is quoted in several places in the following paragraphs.

In *Laborem Exercens*, John Paul II talks about many different facets of work and comments on the value of work, the relationship between technology and work, unions, and the spirituality of work. The scope of the document is extensive and therefore we should try to focus our discussions to the sections on the relationship between technology and work.

Simply stated, the Church believes that work is important, that human work is “key, probably the essential key, to the whole social question...” John Paul II also differentiates between ‘objective’ and ‘subjective’ work. The objective aspect of work is “simply the external aspects of work, the actual job one does, with its necessary tools or machines.” “Work in the subjective sense is something different; it is man himself, man as a worker and the subject of work.”

The concern the Church has is that we, who are supposed to be the subject of work, have become simply another tool in the production of goods. It is important to note that the Church is not against technology or the use of technology in work. The difficulty arises when humankind is brought down to the level of the tools (i.e., labour is just another factor in the equation of profit). John Paul II “insists especially on human work as a sharing in the activity of God the Creator.”

This is foreign to the current way of thinking in the industrial world. This is a wonderful chance to expose students to another way of thinking, a healthier way of thinking, about the world. I think the key to this is the way students think of work – a way of making money. Whereas the Church says work is a way to fulfill our humanity and a way to share in the activity of God.

The use of technology becomes a problem when it is used as a means to displace workers, to increase efficiency and productivity to make more profit. The Church has no problem with the use of technology to increase efficiency and productivity as long as the workers share in the wealth (not just in the monetary sense) brought about by its use.

In the encyclical, John Paul comments on technology, ‘While it may seem that in the industrial process it is the machine that “works” and man merely supervises it, making it function and keeping it going in various ways, it is also true that for this very reason industrial development provides grounds for reproposing in new ways the question of human work.’ And it is this “reproposing” that brings problems - there is a shift in the way we think of people and their relationship to work.

The Church sees technology as an ally, but adds, ‘However, it is also a fact that, in some instances, technology can cease to be man's ally and become almost his enemy, as when the mechanization of work “supplants” him, taking away all personal satisfaction and the incentive to creativity and responsibility, when it deprives many workers of their previous employment, or when, through exalting the machine, it reduces man to the status of its slave.’

When we become “slaves” to technology, then we no longer are the subject of work. We are no longer that important - workers become expendable. However, it is not technology that is the problem, for technology is merely a tool. It is the placement of technology over humankind that is the root of the problem.

To prepare for this activity, the teacher must spend some time reflecting on and praying about the contents of the encyclical. This is key to making this an effective activity. It is important to challenge students' thinking about work, about technology, and about the use of technology in work. However, it can only be achieved through a thoughtful and careful presentation of the issue.

Appendix 3.6.2

Reflecting on Work – Sample Assignment

Present the following text to students in the form of a handout. This text is borrowed from *Laborem Exercens* (in everyday language) by Joseph Donders - http://www.osjspm.org/cst/le_el.htm (the text can be found in Joseph Donders' book entitled *John Paul's Encyclicals in Everyday Language*).

The class can be divided into small groups to facilitate discussion. The text can be divided into sections and each section treated separately. Each section should be read in class with some comments offered by the teacher. The section should then be discussed within the small groups. The teacher should prepare two or three questions for each section. These questions can then be used to direct the discussions in the small groups. The teacher leads class discussion regarding each section.

Some of the questions prepared by the teacher should solicit stories from students regarding their experience or their parents' experience with the effects of the use of technology in the workplace.

This is followed by an assignment to write a reflection paper as homework. This gives students a chance to reflect on the encyclical at a personal level. The reflection paper may be based on one of the following questions and be at least 250 words (in essay format)

- In light of the reading presented during class, has your view on our relationship to technology changed?
- In light of the reading presented during class, has your view of work changed?
- Do you agree with the Church's view on the relationship of technology and work?

During the second hour of the activity (possibly the day after the homework question is assigned), the teacher can solicit contributions from students' reflection papers to spark further discussion regarding this topic. The teacher can also present other material on the same topic.

There are obviously two sides to this topic. It might be useful and insightful to invite two people to speak on each side (possibly a union representative and a senior manager from a local company). They can talk about their views on the relationship between technology and work.

The following is the text borrowed from parts of *Laborem Exercens* (in everyday language). Note that this is not the actual text of *Laborem Exercens*, but a translation to everyday language. Furthermore, this does not represent the whole of the encyclical.

Preface

“Human beings earn their daily bread through work. Through work they contribute to science and technology and to the enrichment of the moral and cultural of their society. By work we mean any human activity, whether manual or intellectual. Made in the image of God, human beings are placed on earth to have power over it. From the beginning they have been called to work. It is work that distinguishes human beings from other creatures. They are the only ones capable of work. Work is something particularly human done in a community of persons, a characteristic that marks and, in a sense, constitutes the very nature of work.”

Work in the Objective Sense: Technology

“Work is – objectively – what a human being does when dominating the earth. Work has been changing during the ages, from domesticating animals and extracting resources from earth and sea, to cultivating the earth, transforming, changing, and using its produce. Agriculture remains vital to economic activity and production. Industry links the earth’s riches with human work, whether physical or intellectual. Today much human work has ceased to be manual; hands and muscles are helped by machinery, by electronics and micro-processing. It may seem that it is the machine that “works,” but it is the human being who works, and who remains the subject of work. Technology is humanity’s ally; it eases our work, it perfects, accelerates, and increases it.”

Technology sometimes becomes almost an enemy, supplanting workers, taking away personal satisfaction, creativity, and responsibility, causing unemployment, or making workers mere slaves of the machine. Technology is definitely covered by the biblical word “subdue the earth” and it has been correctly seen as a basic aid to economic progress, but it also raises many social and ethical questions on how to relate to work and even to each other - questions challenging states, governments, international organizations, and the church.”

Work in the Subjective Sense: The Worker as Subject

“We must pay more attention to the one who works than to what the worker does. The self-realization of the human person is the measure of what is right and wrong. This basic truth has always been the heart of Christian teaching on human work. The ancient world divided people into classes according to the type of work that people did. Manual work was done by slaves and considered to be unworthy of free people. Broadening what the Bible had said and seeing it in the light of the Gospel, Christianity changed this idea. The one who, while being God, became equal to us in all things spent most of his life at the carpenter’s bench, showing that the value of work does not depend on the type of work done, but on the person who is doing the work. Human persons and not what they do determine the dignity of work. This does away with the division of people into classes according to the work they do. Work can be classified and rated, but the measure of the value of any work remains the human being, who is its “subject.” Work is in the first place “for the worker” and not the worker “for work.” Work itself can have greater or lesser objective value, but all work should be judged by the measure of dignity given to the person who carries it out. Work has no meaning by itself; it is always the human being who counts, even if the work done is the most monotonous or alienating.”

A Threat to the Right Order of Values

“This Christian “gospel of work” had to oppose the materialistic and economist thought of the modern age. Work was understood as “merchandise” sold by the workers to their employer, the one who owned everything necessary for production. These nineteenth-century ideas have given way to a more human thinking about work, but the danger of treating work as “merchandise” - or as an impersonal “work force” - remains as long as economics is understood in a materialistic way. It is this one-sided approach that concentrates on work as the prime thing, leaving the worker in a secondary place. This is a reversal of the order laid down in the book of Genesis. The worker is treated as a tool whereas the worker ought to be treated as the subject of work, as its maker and creator. This reversal - whatever other name it gives itself - should be called “capitalism” - an economic and social system that historically has been known as opposed to “socialism” or “communism.” The error of early capitalism can be repeated wherever the worker is treated as a mere means of production, as a tool and not as a subject. To consider work and the worker in the light of humanity’s dominion over the earth goes to the very heart of the ethical and social question. It is in insight that should be applied to all social and economic policy, within each country, but also internationally, to the tensions between East and West, North and South.”

Abridged version (The Encyclicals in Everyday Language) - copyright © 1996 by Joseph G. Donders.

Appendix 3.6.2 (Continued)

Reflecting on Work – Reflection Career Assignment Assessment Tool

Use the following tool to self-assess your career assignment.

Criteria	High	Moderate	Needs Improvement
To what degree am I able to:			
<ul style="list-style-type: none"> read, understand, and use written material from various sources and mediums such as books, magazines, CD-ROMs, and networked resources 			
<ul style="list-style-type: none"> examine and describe personal values, abilities and aspirations 			
<ul style="list-style-type: none"> describe the meaning and purpose of work as a part of life 			
<ul style="list-style-type: none"> relate the concept of vocation to career choice 			
<ul style="list-style-type: none"> provide examples of how work can provide meaning, dignity, fulfillment to the individual and also contribute to the common good 			
What do you think went well in your career assignment?			
What would you do differently in the next assignment?			
What help do you need to improve your work?			

Unit 5: Using Data Structures

Time: 18 hours

Unit Description

This unit focuses on the programming techniques required to store and manipulate data and to solve problems through the development of a database. Each activity develops knowledge and skills that students apply in the culminating challenge of this unit, to develop a database for a school team (e.g., the hockey team or similar organization, consisting of personal data such as player name, position played, jersey number, phone number, goals, and assists). Students examine the structuring of one- and two-dimensional arrays and how data is represented and stored in these structures. They write programs that create lists and tables of data, manipulate the data, and output the result. Sorting and searching techniques are also applied.

Unit Synopsis Chart

Activity	Time	Expectations	Assessment	Task/Strategy
1: Examining Data Structures	1 hour	TFV.03, TF2.05 CGE4f	Communication Knowledge/Understanding	Class discussion Group problem solving
2: Data in Lists	3.75 hours	SP1.03, SP2.02, SP2.10, SP2.14, SP2.15, SP2.16 CGE7h	Knowledge/Understanding Application	Lab exercises
3: Relating Lists	2 hours	SP1.03, SP2.02, SP2.03, SP2.10, SP2.14, SP2.15 CGE5a	Knowledge/Understanding Application	Lab exercises Quiz
4: Data in Tables	3.75 hours	TFV.03, TF2.05, SP1.03, SP2.02, SP2.10, SP2.14, SP2.15 CGE5e	Knowledge/Understanding Application	Lab exercises
5: Sorting Data	3.75 hours	SP1.07, SP2.02, SP2.10, SP2.14, SP2.15, SP2.16 CGE3c	Application	Lab exercises
6: Searching Lists and Tables	3.75 hours	SP2.02, SP2.10, SP2.14, SP2.15, SP2.16 CGE7h	Application	Lab exercises Assignment Unit test

Activity 1: Examining Data Structures

Time: 60 minutes

Description

Students identify the programming techniques (i.e., storage of data in lists and tables, sorting and searching data) that are necessary to program a database. This allows students to write programs that efficiently store and manipulate larger amounts of data than the programs developed in previous units. Array concepts, including element value, index, and bounds, are explored using models. They examine written examples of programs illustrating basic application of a variable array. Students develop thinking skills through analysing programs using variable arrays to input, process, and output data (e.g., class set of marks).

Strand(s) & Learning Expectations

Ontario Catholic School Graduate Expectations

CGE4f - applies effective communication, decision-making, problem-solving, time, and resource management skills.

Strand(s): Theory and Foundation

Overall Expectations

TFV.03 - explain standard control and data structures used in computer programs.

Specific Expectations

TF2.05 - define the structure of one- and two-dimensional arrays and associated concepts (e.g., subscripts, elements, bounds).

Prior Knowledge & Skills

Students:

- understand how data is stored in string and numeric variables (Unit 2);
- can use an operating system to perform tasks such as managing files;
- can use built-in networking functions such as shared files and input/output devices;
- are able to implement a comprehensive backup strategy for files.

Planning Notes

- Review basic syntax of one-dimensional array structures for specific language used (Appendix 5.1.1).
- Prepare notes to illustrate basic structures of declaring arrays, totalling array elements, inputting data from the keyboard or data file, and outputting results to the screen.
- Prepare demonstrations using models (e.g., on chalkboard, paper and pencil, deck of cards), giving students visual representations of data storage and manipulation in arrays.

Teaching/Learning Strategies

The teacher:

- presents visual representations of array structures using models such as a set of numbered envelopes, each containing a word or number;
- assists students in identifying examples of data that can be represented using an array (e.g., names of motel room occupants or post office box holders);
- defines basic structure of a one-dimensional array;
- clarifies differences between the index (1, 2, ...10) and the value of an array element;
- demonstrates, through the development of the code for a simple program, the control structures necessary to store data in an array.

Students:

- in groups of four, brainstorm a description of the data that would be part of a database of a sports team in the school and the programming techniques required (e.g., representing a table of data, sorting data, and searching for matching data), which forms the culminating challenge for this unit;
- manipulate visual models, illustrating storage of data in one-dimensional arrays;
- in pairs, summarize concepts of arrays in their own words and apply the concepts by identifying additional examples of data that can be represented using an array;
- develop, using pencil and paper, the code for a simple program (e.g., storing and displaying the list of classes that they are currently taking).

Assessment & Evaluation of Student Achievement

Evidence of student achievement of the concepts of one-dimensional variable array structures is assessed at the completion of Activity 2. At that time, students have applied the concepts in the development of programs requiring the use of one-dimensional arrays.

Accommodations

The following are ways in which the activity can be adapted to accommodate the student's needs:

- teachers should consult the OSR, IEP, Guidance, and Special Education staff with respect to student exceptionalities;
- provide print copies of on-line and print resources to individual students as appropriate;
- provide support through one-to-one teacher-directed conferencing to ensure understanding.

Resources

Hume, J.N.P. *Problem Solving and Programming in Turbo Pascal*. Toronto: Holt Software Associates Inc., 1994. ISBN 0-921598-19-X

Hume, J.N.P. *Problem Solving and Programming in Turing*. Toronto: Holt Software Associates Inc., 1993. ISBN 0-921598-16-5

Wright, Peter. *Peter Wright's Beginning Visual Basic 6.0*. Birmingham, UK: Wrox Press, 1998. ISBN 1-861001-05-3

Activity 2: Data in Lists

Time: 225 minutes

Description

Students write programs to store, process, and output lists of string or numeric data, using a one-dimensional array to solve problems, such as managing a class set of marks, the names of the elements, or members of a school team, and applying the concepts examined in Activity 1. They move data between an array and a sequential file and use a linear search to locate matching information in the array. They also apply tracing techniques to error-check and debug programs.

Strand(s) & Learning Expectations

Ontario Catholic School Graduate Expectations

CGE7h - exercises the rights and responsibilities of Canadian citizenship.

Strand(s): Skills and Processes

Specific Expectations

SP1.03 - select suitable data structures to represent information;

SP2.02 - incorporate one-dimensional and two-dimensional arrays into computer programs;

SP2.10 - incorporate and maintain internal documentation to a specific set of standards, including author, date, file name, purpose, and explanatory comments of major statement groups;

SP2.14 - trace program execution using manual methods and software debugging tools;

SP2.15 - identify and correct logic, runtime and syntax errors in programs;

SP2.16 - use linear searches and simple sort routines in programs.

Prior Knowledge & Skills

Students:

- have stored and manipulated data in sequential files;
- can apply repetition structures (loops);
- can apply selection structures (if-then);
- are able to incorporate subroutines, functions, and menus;
- have used tracing and debugging techniques.

Planning Notes

- Review sequential file structure and manipulation.
- Choose appropriate examples and models illustrating the use of arrays and files.
- Review data types.
- Prepare sample data files for use with the exercise problems.

Teaching/Learning Strategies

The teacher:

- facilitates a discussion on the steps for using one-dimensional arrays in creating and storing lists of information;
- illustrates, using a program example, the step-by-step logical process for reading a list of information into an array and determining the number of elements in the list;
- solicits responses from students to develop pseudocode for a program using a list (e.g., a list of integers), including array declaration;
- assists students in applying developed tracing and debugging techniques to array problems;
- assigns programming exercises requiring use of arrays.

Students:

- work in pairs to enter and debug program code developed in Activity 1;
- working in pairs, develop code based on the pseudocode developed by the group discussion;
- individually solve problems (Appendix 5.2.1) requiring:
 - a. creating a variable array to store a list of a specific data type;
 - b. determining the number of elements in an array;
 - c. displaying information contained in an array;
 - d. reading data from a sequential file into an array;
 - e. writing data stored in an array to a sequential file;
 - f. manipulating and analysing the data contained in an array (e.g., finding total, mean, smallest, and largest values in an array.);
 - g. searching for specific elements in an array.

Assessment & Evaluation of Student Achievement

Students complete and submit their solution to a selected problem from the exercises. Evidence of students' achievement of the application of one-dimensional variable array structures in the solution of the problem is assessed using a rubric (Appendix 5.2.3). The rubric is distributed to students at the outset of the activity.

Accommodations

The following are ways in which the activity can be adapted to accommodate exceptional students' needs:

- provide print copies of subroutines and vocabulary new to students;
- selectively pair students to assist with developing and debugging code;
- provide a “scaffolded” program, with subroutine heading, etc. included, to help struggling students.

Resources

Stephenson, Chris. *Turing Stuff: A Collection of Teacher-Created Exercises, Projects & Quizzes*. Toronto: Holt Software Associates Inc., 1998. ISBN 0-921598-18-1

Introduction to Visual Basic – Project 12 (Variable Arrays) –

<http://www.moorpark.cc.ca.us/~gcampbell/begVB12.htm>

Arrays in C++ - <http://www.cee.hw.ac.uk/~pjbk/pathways/cpp1/node176.html>

Pascal Tutorial 5C One-dimensional Arrays – <http://www.taoyue.com/tutorials/pascal/pas5c.html>

Activity 3: Relating Lists

Time: 120 minutes

Description

Students write programs that store, retrieve, and process related lists of information (e.g., a class set of student surnames, first names, and marks) using multiple one-dimensional arrays similar to those used in Activity 2. This expands the scope of problems that students can solve to include problems such as displaying the atomic number, name, and atomic mass of an element when the atomic number is specified. Data is transferred from sequential access data files to the arrays and back to the files.

Strand(s) & Learning Expectations

Ontario Catholic School Graduate Expectations

CGE5a - works effectively as an interdependent team member.

Strand(s): Skills and Processes

Specific Expectations

SP1.03 - select suitable data structures to represent information;

SP2.02 - incorporate one-dimensional and two-dimensional arrays into computer programs;

SP2.03 - write programs that use related arrays to store and extract data;

SP2.10 - incorporate and maintain internal documentation to a specific set of standards, including author, date, file name, purpose, and explanatory comments of major statement groups;

SP2.14 - trace program execution using manual methods and software debugging tools;

SP2.15 - identify and correct logic, runtime, and syntax errors in programs.

Prior Knowledge & Skills

Students:

- store and manipulate data in sequential files;
- use repetition structures (loops);
- use selection structures (if-then);
- use subroutines, functions, and menus.

Planning Notes

- Select examples of data that are presented in related arrays (e.g., list names of players found on baseball cards, the team they play for, and the value of their card).
- Prepare models illustrating the use of related arrays (e.g., rows of disposable cups, each row labelled with a name, and each cup in the row with a sequential index).
- Prepare presentation materials illustrating the declaration and use of related arrays in sample programs.
- Prepare sample data files for use with the exercise problems.
- Prepare a quiz (Appendix 5.3.2).

Teaching/Learning Strategies

The teacher:

- introduces the rationale for and structure of related arrays;
- leads discussion with students to develop the steps for creating and storing lists of information in related arrays;
- demonstrates writing of code based on an example of pseudocode developed by students;
- assigns programming exercises requiring the application of related arrays.

Students:

- brainstorm examples of applications for related arrays;
- in small groups, develop criteria for determining when to use related arrays;
- working collaboratively in groups, develop pseudocode for a program that requires use of related arrays (e.g., lists of student names and ages), including array declaration;
- individually complete exercises (Appendix 5.3.1), including:
 - a) creating parallel arrays to store lists of related data of similar or mixed data types;
 - b) determining the number of elements in each array;
 - c) displaying tables showing the information contained in the arrays;
 - d) generating and storing data in related arrays;
 - e) reading data from a sequential file into parallel arrays;
 - f) manipulating and analysing the data contained in the arrays (e.g., finding the name of the youngest or oldest person in the class);
 - g) searching for specific information in the arrays (e.g., finding the names of all students of a specific age or finding the age of a specific student).

Assessment & Evaluation of Student Achievement

Students complete a programming and theory quiz (Appendix 5.3.2). Programming solutions are evaluated for application of related arrays, solution of the program, programming style, and internal documentation.

Accommodations

The following are ways in which the activity can be adapted to accommodate exceptional students' needs:

- selectively pair students to assist with developing and debugging code;
- provide print copies of code for programs similar to those in the exercises;
- provide partially developed programs requiring completion.

Resources

Kaufman. *Problem Solving and Structured Programming Pascal*. Don Mills: Addison-Wesley Publishing Co. Inc., 1995. ISBN 0-201-11736-3

Activity 4: Data in Tables

Time: 225 minutes

Description

Students write programs to store, process, and output tables of string or numeric data using a two-dimensional array. This permits more efficient handling of data than the use of related arrays for tables of consistent data type. They apply the techniques to problems selected from a variety of contexts (e.g., a salary grid or table of measured precipitation).

Strand(s) & Learning Expectations

Ontario Catholic School Graduate Expectations

CGE5e - respects the rights, responsibilities, and contributions of self and others.

Strand(s): Theory and Foundation, Skills and Processes

Overall Expectations

TFV.03 - explain standard control and data structures used in computer programs.

Specific Expectations

TF2.05 - define the structure of one- and two-dimensional arrays and associated concepts (e.g., subscripts, elements, bounds);

SP1.03 - select suitable data structures to represent information;

SP2.02 - incorporate one-dimensional and two-dimensional arrays into computer programs;

SP2.10 - incorporate and maintain internal documentation to a specific set of standards, including author, date, file name, purpose, and explanatory comments of major statement groups;

SP2.14 - trace program execution using manual methods and software debugging tools;

SP2.15 - identify and correct logic, runtime, and syntax errors in programs.

Prior Knowledge & Skills

Students:

- have used repetition structures (loops);
- know how to store and retrieve data stored in sequential files;
- are able to use one-dimensional arrays in solving problems.

Planning Notes

- Review basic syntax of two-dimensional arrays for the specific language.
- Prepare notes to give students basic structures of declaring two-dimensional arrays, moving across individual rows and columns, as well as reading from a file and outputting results.
- Choose examples of problems that require use of two-dimensional arrays in solution.
- Prepare demonstrations through use of models (e.g., on chalkboard, paper and pencil, deck of cards) to give students visual representations of two-dimensional arrays.
- Prepare exercises manipulating two-dimensional arrays.

Teaching/Learning Strategies

The teacher:

- introduces the components of a two-dimensional array;
- assists students in identifying the data structures that are appropriately managed with two-dimensional arrays;
- clarifies the use of row and column labels to assist students' understanding of the role of each index in a two-dimensional array;
- emphasizes restrictions on data types when using a two-dimensional array;

-
- demonstrates development of pseudocode for a program requiring use of two-dimensional arrays (e.g., tables of wind-chill values or stock inventory items including item number, description, quantity in stock, and retail price), including array declaration and the use of nested loop structures;
 - provides examples of programs to illustrate use of tracing and debugging techniques in identifying and correcting program code errors (e.g., incorrect index labelling, subscript out-of-range errors);
 - assigns exercises.

Students:

- work in groups to describe how array concepts can be applied to examples such as battleship; chess;
- in groups, manually trace the execution of a simple program that uses a two-dimensional array;
- in pairs, develop and debug code based on the pseudocode developed as a group;
- complete exercises (Appendix 5.4.1), including:
 - a) creating two-dimensional arrays;
 - b) determining number of rows and columns required by the array;
 - c) declaring array appropriately;
 - d) displaying tables showing the information contained in the array;
 - e) reading data from and to a sequential file using a two-dimensional array;
 - f) manipulating and analysing data contained in arrays (e.g., summing or averaging data contained in rows and columns of numeric two-dimensional array);
 - g) searching for specific information in arrays (e.g., listing first names of all students, given a specified surname).

Assessment & Evaluation of Student Achievement

Students, working in pairs, assess each other's solution to a selected problem from the exercises using a checklist (Appendix 5.4.2).

Accommodations

The following are ways in which the activity can be adapted to accommodate the student's needs:

- provide print copies of additional example programs illustrating the use of two-dimensional arrays;
- use mapping and/or flowcharts to assist students in understanding manipulation of data in two-dimensional arrays;
- selectively pair students to assist with developing programming solutions;
- challenge students to attempt problems requiring multi-dimensional arrays such as recording individual player statistics for multiple teams in a hockey league.

Resources

Arrays in C – <http://www-ee.eng.hawaii.edu/Courses/EE150/Book/chap7/chap7.html>

Arrays in C++ – <http://www.cee.hw.ac.uk/~pjbk/pathways/cpp1/node176.html>

Pascal Tutorial 5D –multi-dimensional arrays - <http://www.taoyue.com/tutorials/pascal/pas5d.html>

Activity 5: Sorting Data

Time: 225 minutes

Description

Students develop algorithms for sorting data using a model. They develop pseudocode and write programs, which use simple sorting algorithms to organize string and numeric data in one- and two-dimensional arrays, using the array manipulation techniques developed in Activity 4 (e.g., calculating the median of a set of data using a bubble sort).

Strand(s) & Learning Expectations

Ontario Catholic School Graduate Expectations

CGE3c - thinks reflectively and creatively to evaluate situations and solve problems.

Strand(s): Skills and Processes

Specific Expectations

SP1.07 - solve the same problem using various tools (e.g., a calculator and a computer program, a sort program and a spreadsheet/database/word processor sort function);

SP2.02 - incorporate one-dimensional and two-dimensional arrays into computer programs;

SP2.10 - incorporate and maintain internal documentation to a specific set of standards, including author, date, file name, purpose, and explanatory comments of major statement groups;

SP2.14 - trace program execution using manual methods and software debugging tools;

SP2.15 - identify and correct logic, runtime, and syntax errors in programs;

SP2.16 - use linear searches and simple sort routines in programs.

Prior Knowledge & Skills

Students:

- use loop structures to manipulate data in one-dimensional arrays;
- use a random number generator;
- manipulate data using one-dimensional arrays;
- structure programs using procedures/sub programs.

Planning Notes

- Review the basic algorithms for sorting data using a computer (Appendix 5.5.1).
- Review the ASCII chart to identify problems students may have when dealing with string variables.
- Assemble materials to be used by students in developing sorting algorithms.
- Prepare exercises requiring the development of programs to apply and compare at least two different sorting methods.

Teaching/Learning Strategies

The teacher:

- introduces the use of a model for the development of sorting algorithms (Appendix 5.5.2);
- facilitates a discussion how to evaluate the sorting algorithms they have developed;
- presents at least two basic, standard algorithms for sorting data using a computer, (Appendix 5.5.1), for student comparison;
- assigns exercises.

Students:

- work in groups to use a model to develop algorithms for sorting data stored in an array (Appendix 5.5.2);
- develop pseudocode for the sorting algorithms they have developed;
- examine Internet-based simulations of sorting algorithms;
- develop program code for a standard sorting algorithm;
- completes exercises, including:
 - a) creating one-dimensional arrays to store lists of data;
 - b) sorting the data in ascending or descending order using at least two algorithms;
 - c) displaying results of the sort;
 - d) comparing the results of a program incorporating a sorting algorithm to that of a spreadsheet.

Assessment & Evaluation of Student Achievement

Students assess their solution to a selected programming problem using a checklist.

Accommodations

The following are ways in which the activity can be adapted to accommodate exceptional students' needs:

- provide examples at levels of programming skills and experience;
- students requiring a challenge and/or enhancement of their program can attempt extension activities involving additional sorting algorithms;
- provide extra time and one-to-one teacher support;
- select problems to allow for both remediation and enrichment.

Resources

Carter, John. *Problem Solving in Pascal*. Toronto: Addison-Wesley Publishers Limited, 1989, pp. 343, 350. ISBN 0-201-11215-9

Sorting Algorithms – http://www-lsi.upc.es/~rbaeza/handbook/sort_a.html

Animation of Sort Algorithms – <http://blackcat.brynmawr.edu/~spoonen/JavaProject/sorter.html>

Bubble Sort – <http://www-ee.eng.hawaii.edu/Courses/EE150/Book/chap10/subsection2.1.2.2.html>

Bubble Sort – http://www.cs.princeton.edu/~ah/alg_anim/gawain-4.0/BubbleSort.html

Insertion Sort Demonstration – <http://www.cs.orst.edu/~minoura/cs162/javaProgs/sort/InsertSort.html>

Activity 6: Searching Lists and Tables

Time: 225 minutes

Description

Students develop algorithms for simple searching routines and apply them to problems (e.g., locating the name of a group in a database of albums and artists). They also complete exercises and write programs applying at least two search algorithms. They complete the culminating challenge presented in Activity 1 – programming a database for a sports team in the school. The database requires the application of techniques for manipulating and sorting data learned in the previous activities.

Strand(s) & Learning Expectations

Ontario Catholic School Graduate Expectations

CGE7h - exercises the rights and responsibilities of Canadian citizenship.

Strand(s): Skills and Processes

Specific Expectations

SP2.02 - incorporate one-dimensional and two-dimensional arrays into computer programs;

SP2.10 - incorporate and maintain internal documentation to a specific set of standards, including author, date, file name, purpose, and explanatory comments of major statement groups;

SP2.14 - trace program execution using manual methods and software debugging tools;

SP2.15 - identify and correct logic, runtime, and syntax errors in programs;

SP2.16 - use linear searches and simple sort routines in programs.

Prior Knowledge & Skills

Students:

- use loop structures to manipulate data in one-dimensional arrays;
- describe basic structure for one-dimensional arrays;
- structure programs using procedures/sub-programs.

Planning Notes

- Prepare exercises for students to compare, program, and analyse at least two searching methods;
- Prepare the unit test.

Teaching/Learning Strategies

The teacher:

- directs students to Internet resources illustrating the algorithms for searching arrays for matching data;
- assigns exercises requiring searching lists and tables of data.

Students:

- working in groups, use a model to create at least two algorithms for searching an array for matching data;
- examine Internet-based simulations of searching algorithms;
- create pseudocode for a searching algorithm;
- write and debug code for a searching algorithm;
- complete exercise problems (Appendix 5.6.2) which require:
 - a) creating arrays to store data of the same type;
 - b) searching matching data using at least two algorithms;
 - c) displaying results of search;
 - d) comparing efficiency of searching algorithms.
- complete a summative programming assignment (Appendix 5.6.3).

Assessment & Evaluation of Student Achievement

- Students submit their solution to the summative assignment to be assessed using a rubric (Appendix 5.6.4).
- Students complete a unit test (Appendix 5.6.5).

Accommodations

The following are ways in which the activity can be adapted to accommodate exceptional students' needs:

- provide parts of subroutines or parts of code to aid struggling students;
- provide print copies of pseudocode for searching routines;
- challenge students to investigate additional searching algorithms.

Resources

Carter, John. *Problem Solving in Pascal*. Toronto: Addison-Wesley Publishers Limited, 1989, pp. 334-337. ISBN 0-201-11215-9

Searching Algorithms – http://www-lsi.upc.es/~rbaeza/handbook/search_a.html

Appendix 5.1.1

Array Concepts

Arrays : A table of values of the same type and a fixed size. The table may have one or more dimensions.

Components : The data within the array.

Index : The position of a component in an array. The index must be of an ordinal type.

Example: Given the array declaration in Pascal OR Basic
 MARKS : ARRAY[1..10] OF INTEGER; DIM MARKS(1 TO 10) AS INTEGER

If we were to read into the array the following data

1	2	3	4	5	6	7	8	9	10
72	44	66	87	54	85	92	65	71	63

The components are the marks themselves.

The components can be accessed with MARKS[1] or MARKS[5]. In Pascal

OR MARKS(1) or MARKS(5) in Basic

The indices are 1,2,3...10

Therefore, MARKS[3] or MARKS(3) has a value of 66

MARKS[7] or MARKS(7) has a value of 92

Manipulating Values in Arrays

Assuming the following declaration:

In Pascal	In Basic
VAR MARKS : ARRAY [1..10] OF INTEGER; I, J : INTEGER; TOTAL : INTEGER; AVERAGE : REAL;	DIM MARKS(1 to 10) as INTEGER DIM I, J as INTEGER DIM TOTAL as INTEGER DIM AVERAGE as REAL

Suppose a file contained ten records, we could read each value into the arrays MARKS, and NAMES with the following:

In Pascal	In Basic
FOR I := 1 TO 10 DO READLN(INFILE,MARKS[I]);	FOR I = 1 TO 10 INPUT #1, MARKS(I) NEXT I

We could find the average with the statement:

In Pascal	In Basic
TOTAL := 0; FOR J := 1 TO 10 DO TOTAL := TOTAL + MARKS[J]; AVERAGE := TOTAL / I;	TOTAL = 0 FOR I = 1 TO 10 TOTAL := TOTAL + MARKS(J); NEXT I AVERAGE = TOTAL / I

We could print out the records with the statement:

In Pascal	In Basic
FOR I := 1 TO 10 DO WRITELN(MARKS[I]:8); WRITELN('Average = ',AVERAGE:5:2);	FOR I = 1 TO 10 PRINT#1, MARKS(I) NEXT I PRINT #1, AVERAGE

Appendix 5.2.1

Data in Lists – Sample Lab Exercises

1. To use the “speed dial” feature of your home phone you must first store each of the phone numbers that you wish to use. Each phone number is accessed by entering the speed-dial number, from 0 to 9, used for storing the number. Write a program that will help you remember who’s phone number you have stored with each speed dial number. Your program must use an array to store the list of names. The user enters a number from 0 to 9 and is shown the name of the person that would be called if you entered that speed dial code into the phone. Store the data from the array in a sequential file.
2. Write a program that allows the user to enter up to 20 numbers to be stored in an array. When the user has finished entering numbers, calculate and display the sum of the numbers, the mean (arithmetic average) of the numbers, and the largest and smallest values entered.
3. Create a program that reads a list of no more than 100 names from a sequential file into an array. The names are stored in the file in the form “last_name, first_name.” Allow the user to display the list; add, modify, and delete names; and save the modified list in a sequential file with the same name.
4. Write a program to simulate rolling two dice together 100 times. Count the number of times each result (i.e., total of the two dice) occurs and print out the results. Note that rolling two dice together is not the same as picking a number from 2 to 12!! (If you really turn on your problem-solving skills, this program can be most easily done WITHOUT AN IF STRUCTURE!!)

Appendix 5.2.

Rubric for Assessment of Solutions to One-dimensional Array Problems

Achievement Category	Level 1 (50 – 59%)	Level 2 (60 – 69%)	Level 3 (70 – 79%)	Level 4 (80 – 100%)
Understanding of array concepts (K/U)	- provides limited differentiation between indexed and non-indexed variables	- sometimes differentiates between indexed and non-indexed variables	- usually differentiates between indexed and non-indexed variables	- demonstrates an understanding of efficiency of data representation in lists
Identification of data (K/U)	- rarely identifies characteristics of data best represented as lists	- identifies some data best represented as lists	- usually identifies data best represented as lists	- clearly differentiates data best represented as lists
Problem Solving (A)	- approaches problems in a trial and error manner	- develops incomplete and/or insufficiently detailed plan	- properly plans and details solution in a stepwise manner	- properly plans highly efficient solution to problem
Documentation (C)	- documentation communicates appropriate information in a limited manner	- documentation provides some description and identification of programming logic	- correctly and appropriately comments on code and variables	- provides complete documentation (help screens and/or external documentation)
Programming Style (C)	- rarely uses proper style (indentation, separation of program structures)	- usually uses proper style	- consistently uses proper style	- consistently uses proper conventions and style
Declaration of array variables (A)	- improperly declares arrays in a limited manner	- declares arrays and uses some appropriate variable types	- declares arrays using appropriate data types and index values	- code reflects efficient use of arrays
Use of indices (A)	- rarely uses indices	- partially implements use of indices in programming structures	- consistently uses indices correctly in programming structures	- uses indices correctly with a high degree of programming structures
Solution of problem (A)	- program rarely solves problem	- program is functional but fails limit testing	- program adequately solves problem	- program provides exemplary solution
User Interface (A)	- user interface rarely communicates with user	- confusing and/or incomplete user interface	- user interface is intuitive and user-friendly	- user interface consistent with industry standards

Note: A student whose achievement is below level 1 (50%) has not met the expectations for this assignment or activity.

Appendix 5.3.1

Relating Lists - Sample Lab Exercises

1. Create a program that reads a list of names from a sequential file into an array, storing the name, and another array corresponding to their age. The arrays must be organized so that there is a 1-to-1 correspondence between the indices of the name and age. When the user enters an age, the names of all the people of that age should be displayed. The user should also be able to add, delete, and modify the names and ages entered.
2. Create a program that creates and administers a true/false quiz. Questions are read from a data file into one array and a related array is used to store the answers, either T or F, for each question. When the user takes the test, questions are presented in random order, without repetition. Keep track of the number of questions answered and the number of correct answers. When the user has finished the quiz, display the mark as a percentage.

Appendix 5.3.2

Quiz for One-dimensional Arrays

Part 1 – Answer the following questions on a separate piece of paper.

1. Explain the difference between the index and the element of an array.
2. Describe how data is linked in related arrays.
3. Give an example of a set of data that is best represented by related arrays.

Part 2 – Create a program to solve the following problem. Make sure your code is properly documented and structured. Print one copy of your code to hand in.

The sample data file “quiz1.dat” contains a list of students in a class and their marks for the class.

Read the marks into two related arrays.

Using the values stored in the arrays, determine and display the class average and the name of the student with the highest mark.

Appendix 5.4.1

Two-dimensional Arrays - Sample Lab Exercises

1. This table contains the average weekly precipitation for four Ontario Cities.

Month	Precipitation (cm)			
	London	Kingston	North Bay	Dryden
January	12.4	13.5	12.1	12.6
February	8.2	9.5	8.0	7.5
March	14.9	13.5	12.9	15.3
April	30.0	35.5	40.2	37.7
May	24.5	26.1	23.4	24.0
June	30.3	4.6	11.4	21.6

- Create a data file, containing the above data, using a spreadsheet and saving the data as a comma delimited file.
 - Declare the two-dimensional array for the above data called PRECIPITATION.
 - Read the data from the data file into the array PRECIPITATION.
 - Print out the precipitation data in a neat table with headers and row titles.
 - Calculate and display:
 - the average weekly precipitation for North Bay;
 - the average precipitation for April;
 - the total precipitation for all areas for the first six months of the year;
 - the city with the largest precipitation for June;
 - the month with the smallest precipitation in London.
2. Write a program that reads the police salary grid into a two-dimensional array from a data file you have prepared. Negotiators have asked you to project new pay grids based on various increases. Use 1.5%, 2.5%, and 4% increases in pay as your test data. Calculate the new pay increases across the pay grid. Output the original grid and the new grids with pay increases. The grid below is used to pay police officers:

Police Pay Schedule				
		Rank		
		Lieutenant	Sergeant	Constable
	0	30000	28750	26500
Y	1	32000	30540	28750
E	2	34000	32400	29900
A	3	37300	34100	31200
R	4	39100	36200	33300
S	5+	40000	38000	35000

Appendix 5.4.1 (Continued)

The police Commission is also concerned with the cost of the raise in pay. The following is a table indicating how many employees are in each category.

Number of Employees in Each Category				
		Constable	Sergeant	Lieutenant
	0	89	90	55
Y	1	90	36	47
E	2	45	23	20
A	3	73	41	42
R	4	91	62	33
S	5+	12	80	52

Find the total cost to the Police Commission if any of the above pay increases was granted. Display your results in an appropriate format.

Appendix 5.4.2

Checklist for Assessment of Problem 2

Program Title: _____ Programmer: _____
Assessed by: _____

User Interface

(circle choice)

1. Program is able to read data from a file. Y / N
2. Data is aligned in columns. Y / N
3. User can easily input percentage increase. Y / N
4. Input is checked for acceptable value. Y / N
5. Appropriate on-screen prompting is displayed. Y / N

Problem Solution

6. Pay increases have been correctly calculated. Y / N
7. Total cost of pay increase is correctly calculated. Y / N

Program Structure

8. Program is appropriately divided into subroutines. Y / N
9. Program header contains required information. Y / N
10. Internal documentation is complete. Y / N
11. Program uses a two dimensional array. Y / N
12. Variable names are appropriate. Y / N

Appendix 5.5.1

Sorting Data

There are many different types of sorting routines. Below is the pseudocode for two different sorts.

Selection Sort

In a selection sort, we start at one end of the array looking for the largest or smallest depending on whether the sort is in ascending or descending order. Once found, we can place it in its proper spot and continue searching through the remaining elements in the array. Once an element has been swapped, it is never moved again. One disadvantage to this type of sort is that it must perform N^2 passes through the array regardless of the original order.

```
for i = the beginning of the array to the end
    'Look for the smallest (or largest) element and put it in the ith element
    for j = the ith element to the last element
        if the jth element is smaller than the ith element
            smallest = j
    swap the jth element with the ith element after going through the array
```

Bubble Sort

In a bubble sort, adjacent values are compared and exchanged if they are not in order. It uses a boolean variable "SORTED" which determines if the list is sorted and stops execution if found to be in order. Although many more swaps are performed in this sort, it is especially efficient if only a few elements have been added to the array, since the sort is a "smart" sort and stops when the boolean flag becomes true.

```
procedure sort
set sorted to false
set max to top
while not sorted and top > 1
    set sorted to true
    for passes = 1 to top - 1
        if element at i > list at i + 1
            exchange the ith element with the ith + 1 element
    set top to top - 1
```

Appendix 5.5.2

Developing a Sorting Algorithm

Purpose

To use a model to develop an algorithm for sorting data stored in a one-dimensional array.

Description

A series of labelled disposable cups represent the elements of an array. A small, folded piece of paper is placed in each cup. Each piece of paper has a number, letter, or word written on it. Students, working in small groups, sort the data (pieces of paper) using methods that are available within a computer program. A description of the algorithm developed, pseudocode for the algorithm, and code are developed.

Method

1. Label each of eight disposable cups with a name and a number similar to the labelling of elements in an array (e.g., cup(1), cup(2), cup(3), etc.). Place the cups in a line in order.
2. Cut a piece of paper into eight pieces, approximately 3 cm x 6 cm. Write the numbers from 1 to 8 on the pieces of paper and fold each in half so that the number is hidden.
3. Mix up the pieces of paper and place one piece in each cup.
4. Using the following rules, develop a method for sorting the numbers on the paper in order so that the smallest number is found in cup(1) and the largest is in cup(8) and all others are in sequence. Use brainstorming techniques with your group.
 - a. Only two numbers may be removed from the cups at any one time.
 - b. A number being viewed may be placed only in the cup from which it was removed or a cup that does not contain a number currently being viewed.
 - c. Papers must remain folded so that the number is hidden when the paper is in the cup.
 - d. Additional cups may be used, but they must be labelled.
 - e. The cups must remain in order at all times.
5. Experiment with different suggested methods until you have found a simple method to sort the data that can be described in a few, repeatable steps.
6. Record the steps in the sorting method you have developed (the algorithm). If additional cups are used, their use must be described.
7. Test the steps by having one person read the steps one at a time and another person follow the instructions. Don't make things up – follow the instructions exactly!
8. Test your algorithm with different data (e.g., when one number is repeated).
9. Modify your algorithm based on what you discover during testing.
10. Individually, write pseudocode for the algorithm.
11. Compare your pseudocode with that of your group members.
12. Write code for the algorithm. Don't forget to include variable declarations and variable initialization.
13. Test your code using various data sets, including data in which numbers are repeated and string data (words).

Teacher's Note: Students may create a variety of algorithms. Ask each group to demonstrate their algorithm. As a group, evaluate each algorithm for efficiency (sorting in the fewest steps) and ease of programming. You may wish to describe the algorithm for the Two-Buffer sort and Bubble sort and allow students to experiment with them using the cups.

Appendix 5.5.3

Sorting Data - Sample Lab Exercises

1. Write a program that finds the median of a set of numbers input by the user. In order to find the median you must first sort the numbers in ascending or descending order. If there are an odd number of numbers then the median is the middle number in the list. If there is an even number of numbers, then the median is found by averaging the two middle values.
2. Using the data file "PRECIP.DAT" from problem 1 of Appendix 5.4.1, read the data into a two-dimensional array of 6 rows and 6 columns. Calculate the average precipitation for each month (rounded to 1 decimal place) and store the average in the 6th column. Sort the rows in ascending order by average rainfall and display the resulting table.
3. Repeat problems 1 and 2 using a spreadsheet to store and sort the data. Write a brief report comparing the results achieved by your program and the spreadsheet. Describe how the sort function is used in the spreadsheet and any limitations to its use that you found.

Appendix 5.5.4

Checklist for Assessment of Problem 1

Program Title: _____ Programmer: _____

Assessed by: _____

User Interface

(circle choice)

- | | |
|---|-------|
| 1. Data is easily entered. | Y / N |
| 2. Appropriate prompting is present. | Y / N |
| 3. Input is screened for acceptable values. | Y / N |

Problem Solution

- | | |
|--|-------|
| 5. Median is correctly calculated for odd data numbers. | Y / N |
| 6. Median is correctly calculated for even data numbers. | Y / N |

Program Structure

- | | |
|---|-------|
| 7. Program is appropriately divided into subroutines. | Y / N |
| 8. Program header contains required information. | Y / N |
| 9. Internal documentation is complete. | Y / N |
| 10. Program uses a sorting algorithm. | Y / N |
| 11. Variable names are appropriate to the problem. | Y / N |

Appendix 5.6.1

Searching Lists and Tables

Sequential Search

In a sequential search, we start at the beginning of the array and search one element at a time until the element is found. A Boolean variable “found” is used to allow the search to stop if the element has been located. This search can be inefficient and time consuming, since it must go through the array one element at a time.

Pseudo Code for sequential search

```
Set found = false
Set Index to 1
Input search item
While not found and not at the end of the array
    If list at index = search item
        Found = true
        Location = index
    Endif
End while loop
```

If found = false, output that item was not found

If found = true, output location item was found at.

Binary Search

If your array is in order, then a binary search can be performed and is far more efficient than a sequential search. The basic premise is to divide the array in half and look to see if the item we are searching for is in the top half or the bottom half. We continue cutting the remaining array in half, looking to see which half to search through until the element is found or when the top overlaps the bottom meaning that the item is not found.

Pseudo Code for binary search

```
Set bottom to 1 and top to MAX elements
Set Found to false
Input item searching for
While bottom <= top and not found
    Middle = (bottom + top) / 2
    If list at middle = item
        Found = true
        Location = middle
    Else
        If list at middle < item ‘ discard the bottom half
            bottom = middle + 1
        else
            top = middle - 1 ‘ discard the top half
        endif
    endif
end while loop
```

If found = false, output that item was not found

If found = true, output location item was found at location middle

Appendix 5.6.2

Searching Lists and Tables - Sample Lab Exercises

Written

Answer the following questions:

1. When would the sequential search be more efficient?
2. Why does the binary search require that the array be sorted?
3. Assume that an array contains the following data:
23 34 43 45 52 59 70 75 77 86
Trace the binary search for the following items
89 59 40 75
6. What changes would you need to make to the sequential search if the array is to be sorted in descending order?
7. What changes would you need to make to the binary search if the array is to be in descending order?
8. What is the maximum number of passes necessary to perform a binary search on an array of 100 elements?

Programming Problems

1. Store 20 random numbers in the range 1 to 100 in an array. Display the numbers in the array. Sort the list using the bubble sort and write a function "BINSEARCH" to perform a binary search for a requested value.
2. Create a data file containing a list of at least fifty albums and the artist or group that recorded it. Write a program that allows the user to find the name of the album when they enter the name of the artist or group. Your program must use a binary search.

Appendix 5.6.3

Summative Assignment for Unit 5 – Using Data Structures

You are to create a database for a school team including appropriate player and statistical information. For example, if you choose the hockey team, your data should include each player's first name, surname, position played, height, weight, shoots left/right, plus/minus record, games played, goals, and assists.

The application should allow the user to:

- a) display the team data in alphabetical order by last name;
- b) add new players;
- c) edit the information for a player;
- d) delete players;
- e) find and display an entry based on a search by any field.

Appendix 5.6.4

Rubric for Assessment of Summative Assignment

Achievement Category	Level 1 (50 – 59%)	Level 2 (60 – 60%)	Level 3 (70 – 79%)	Level 4 (80 – 100%)
Solution of Problem (K/U)	- program rarely solves problem	- inefficient algorithm and/or program structure	- program fully solves problem	- program includes additional functionality
Documentation (C)	- documentation rarely communicates appropriate information	- documentation lacks sufficient description and identification of programming logic	- correctly and appropriately uses program headers and comments	- provides additional documentation (help screens and/or external documentation)
Programming Style (A)	- rarely uses proper style (indentation, separation of program structures)	- usually uses proper style	- consistently uses proper conventions and style	- always uses proper conventions and style
Application of sorting algorithm (A)	- program rarely sorts data	- program uses two arrays for sorting	- program uses bubble sort	- more efficient sorting algorithm used
Application of search algorithm (A)	- program rarely locates matching values	- program uses linear search	- program uses binary search	- program uses more efficient search
User Interface (A)	- user interface rarely communicates with user	- confusing and/or incomplete user interface	- user interface is intuitive and user-friendly	- user interface consistent with industry standards

Note: A student whose achievement is below level 1 (50%) has not met the expectations for this assignment or activity.

Appendix 5.6.5

Unit Test

Part 1 - Answer the following questions on the paper provided: (13 marks)

- (4) Give an example of a set of data that is most appropriately presented using related arrays and best represented by a two-dimensional array. Explain characteristics of the data that affected your choice.
- (4) Write the pseudocode for a two-buffer sort to sort a list of fifteen numbers.
- (2) Explain why you might choose to use a bubble sort in your program rather than a two-buffer sort.
- (3) Describe the algorithm for a binary search.

Part 2 – Programming (15 marks)

Save your program in your own directory on the fileserver as “test1”. Be sure to save your program frequently. Proper programming techniques and documentation must be used.

The samples directory contains a file that contains a list of items in stock at the local hardware store, consisting of the item number and description (e.g., A-90321, Screwdriver). Write a program that reads the data into a two-dimensional array and allows the user to display the data sorted in ascending order by item number or name. Also, allow the user to enter the item number and display the matching description.