NCCA

An Chomhairle Náisiúnta Curaclaim agus Measúnachta
National Council for Curriculum and Assessment

Programming and Coding

Draft Specification for Junior Cycle Short Course

October 2013

Contents

Programming and Coding

Introduction to junior cycle

Junior cycle education places students at the centre of the educational experience, enabling them to actively participate in their communities and in society, and to be resourceful and confident learners in all aspects and stages of their lives. Junior cycle is inclusive of all students and contributes to equality of opportunity, participation and outcome for all.

Junior cycle allows students to make a strong connection with learning by focusing on the quality of learning that takes place and by offering experiences that are engaging and enjoyable for them, and relevant to their lives. These experiences are of a high quality, contribute to the physical, mental and social wellbeing of learners, and where possible, provide opportunities for them to develop their abilities and talents in the areas of creativity and enterprise. The student's junior cycle programme builds on their learning in primary school. It supports their further progress in learning. It helps students to develop the learning skills that can assist them in meeting the challenges of life beyond school.

Rationale

Computer Science is present in every aspect of modern society. Correctly functioning software systems allow airplanes to fly from one city to another, give out money at the ATM and diagnose the level of glucose in your blood. Fundamental understanding of how software and computers operate and relate to everyday life is an increasingly important area of learning for students. Problem solving and computational thinking skills are developed in this course as students build and create software projects using their own ideas and imagination. The course also offers students who may be interested in future studies in computer science and software engineering a deeper insight into these areas.

Aim

The course aims to develop the student’s ability to formulate problems logically, to design, write and test code, to develop games, apps, animations and websites and, through these learning activities, to learn about computer science.

Links

Statements of learning (SOL)

Statement	Examples of related learning in the course
The student devises and evaluates strategies for investigating and solving problems using mathematical knowledge, reasoning and skills. SOL 17	Problem solving and computational thinking are central to this course. Students use their mathematical knowledge, skills and understanding when figuring out, evaluating and implementing solutions to particular problems.
The student describes, illustrates, interprets, predicts and explains patterns and relationships. SOL 16	Students interpret and describe patterns and relationships as they solve problems and create projects using algorithms and programming languages.
The student brings an idea from conception to realisation. SOL 23	Students engage in brainstorming and planning activities, move on to the design, development and test phases, culminating in the creation of a project solution to a particular problem.

Literacy and numeracy

Students develop and improve their numeracy skills by actively engaging in problem-solving activities and exploring mathematical and computational ideas. As students create programs, they learn core computational concepts such as iteration and conditionals and mathematical concepts such as variables and random numbers. They learn how to think algorithmically and logically, how to abstract ideas and how to describe patterns and relationships during tasks and projects.

Students develop literacy skills through discussion during class activities that include offering opinions, making oral presentations, writing reports and reflecting on the work in their personal learning journals. Students develop strategies for organising information so that they can understand it, incorporate it into their work and improve their capacity to search for information from different sources.

Other key skills

The six key skills are embedded in the learning outcomes. Below are some examples of learning activities through which the key skills can be developed.

Key skill	Key skill element	Student learning activity
Being Creative	Implementing ideas and taking action	Students brainstorm and generate ideas for design and implementation of solutions and projects.
Communicating	Discussing and debating	Students discuss ideas, evaluate the pros and cons of different approaches, and propose possible solutions. They report on projects and provide feedback to others.

Managing information and thinking	Thinking creatively and critically	Students explore new and different ways of answering questions and solving problems. They use a variety of tools to access, manage and share information such as flow-charts, design documents, code documentation and bug lists.
Managing myself	Setting and achieving personal goals	Students take responsibility for personal learning by setting goals and seeking help when necessary from classmates, the teacher or other appropriate sources, and by reflecting on the feedback they receive.
Staying well	Being safe	Students become aware of the wellness, health and safety issues associated with working with computers, and of practical ergonomic issues regarding the use of computers.
Working with others	Co-operating	Students develop good working relationships with others and appreciate the value of respect and cooperation in reaching both collective and personal goals. They learn to appreciate diverse talents and how to engage in collaborative work.

Course Overview

Strand 1	Computer science introduction - CSI
Strand 2	Let’s get connected
Strand 3	Coding at the next level
Strand 4	Problem solving in the real world

Teamwork is encouraged throughout all four strands. Students should collaborate, peer-explain, seek feedback, provide feedback and reflect on their work. Practical, hands-on learning activities should be in evidence across all strands of the course. Theoretical concepts can be reinforced through practical work and projects.

Free and open-source software should be used where practical, both to make software tools as widely available to students as possible and so that students have the opportunity to examine the source code of the tools they use.

The learning outcomes in this short course are aligned with the Level Indicators for Level 3 of the National Framework of Qualifications (Appendix 1).

The course has been designed for at least 100 hours of student engagement.

Expectations for students

With the publication of the specification online, examples of student work will be used to illustrate the expectations for students in the short course. These examples will be related directly to a learning outcome or groups of learning outcomes. They will be annotated, indicating whether the work is in line with, ahead of, or behind expectations for students.

In the case of short courses in new areas of learning, such as Programming and Coding, some indicative examples of student work will be generated to guide teachers and students during the introductory years of the course.

Strand 1 Computer science introduction – CSI

Students learn about…	Students should be able to …
My digital world: The importance of computers in modern society and my life	1.1 present and share examples of what computers are used for and discuss their importance in modern society and in their lives 1.2 describe the main components of a computer system (CPU, memory, main storage, I/O devices, buses)
Being a coder - step by step: How to start programming and develop basic algorithms	1.3 explain how computers are devices for executing programs via the use of programming languages 1.4 write code to implement algorithms 1.5 test the code 1.6 develop appropriate algorithms using pseudo-code and/or flow charts 1.7 discuss and implement core features of structured programming languages, such as variables, operators, loops, decisions, assignment and modules 1.8 evaluate the results in groups of two or three

Strand 2 Let’s get connected

Students learn about …	Students should be able to …
Making connections: Computers are communication devices	2.1 discuss the basic concepts underlying computer networks 2.2 describe how data is transported on the Internet and how computers communicate and cooperate through protocols such as HTTP 2.3 build web pages using HTML and CSS 2.4 explain how search engines deliver results
Bits and bytes: How computers store data	2.5 explain how computers represent data using 1’s and 0’s 2.6 investigate how drawings and photos are represented in computing devices

Strand 3 Coding at the next level

Students learn about…	Students should be able to…
More advanced concepts in programming and computational thinking	3.1 creatively design and write code for short programming tasks to demonstrate the use of operators for assignment, arithmetic, comparison, and Boolean combinations 3.2 complete short programming tasks using basic linear data structures (e.g. array or list) 3.3 demonstrate how functions and procedures (definition and call) capture abstractions 3.4 describe program flow control e.g. parallel or sequential flow of control – language dependent
Documentation and code analysis	3.5 document programs to explain how they work 3.6 present the documented code to each other in small groups 3.7 analyse code to determine its function and identify errors or potential errors

Strand 4 Problem solving in the real world

Students learn about…	Students should be able to…
Real world problems: Computer Science inspiring me	4.1 identify a topic or a challenge in computer science that inspires them 4.2 conduct research on the topic/challenge
Putting the pieces together: Build a final software project that incorporates concepts learnt in the previous strands	4.3 work in teams of two or three and decide on a topic or challenge on which to build a final software project 4.4 brainstorm ideas in the requirements-gathering phase 4.5 discuss aspects of user-interaction design for the project 4.6 design, implement and test a solution 4.7 document team contributions and document the code

	4.8 present to peers for feedback 4.9 assess the feedback 4.10 based on feedback, complete the software project and present a convincing argument for the final proposal to their peers

Assessment and certification

Assessment and Programming and Coding

This short course supports a wide variety of approaches to assessment. Some learning outcomes lend themselves to once-off assessment, others to assessment on an ongoing basis as students engage in different learning activities such as discussing, explaining, figuring out, researching, making decisions about or reflecting on different aspects of programming and coding. In these contexts, students with their teachers reflect upon and make judgements about their own and others' learning by looking at the quality and features of particular pieces of work. They plan the next steps in their learning, based on feedback they give and receive. Ongoing assessment can support the student in their learning journey and in preparing for the assessment related to the certification of the short course.

Assessment for Certification

Assessment for certification will be school-based. There are two assessment tasks involved: Project Work and a Final Project.

The tasks will be weighted as follows

Project Work 60% (20% each for Strands 1, 2 and 3)	Final Project 40% (Strand 4)

a) Project Work (60%)

Project work is a feature of the learning activity in each strand of the course. The project work undertaken in Strands 1-3 culminates in the final project, which is the focus of Strand 4. The ongoing project work can include reports/documentation, presentations, self and peer-assessment reports, audio or video recordings and interviews. Assignments in programming and coding are also a central feature of assessment of this course. These are a mix of short, focused assignments to give students initial experience of using various programming constructs and longer, more open-ended assignments to consolidate multiple concepts and foster creative problem solving. Students have different opportunities to demonstrate their learning in each strand through assessments based on one or more learning outcomes in the strand and these assessments can be agreed in consultation with the students.

The nature of the project work for Strands 1-3 is set out below

Strand 1 Computer Science Introduction – CSI (20%)
Students will implement a series of short programming tasks. They will involve writing original code implementing algorithms, testing the code, and developing appropriate algorithms using pseudo-code and/or flow charts. Some of the tasks must be completed in teams.

Strand 2 Let's get connected (20%)
Students will demonstrate their understanding of how the Internet works and describe what happens when a user requests a web page in a browser. They will create four basic web pages to demonstrate their understanding of HTML, CSS and hyperlinks.

Strand 3 Coding at the next level (20%)
Students will implement a series of short programming tasks on advanced topics. Some tasks must be completed in teams.

b) Final Project (40%)

The nature of the final project, *Problem solving in the real world*, in Strand 4 is set out below:

Putting the pieces together (40%)
Students identify a topic or challenge in computer science that interests them. They research this and develop a final software project of their choice in teams of 2 or 3. They will gather requirements, design, implement and test the system. They will document their work and their code and present the project to their peers for review. They will reflect on feedback. They will provide feedback on other students' projects.

Features of quality

Features of quality related to student work on the project work and final project are set out below. In general terms, these can be used by students and by teachers to support their discussions about and judgements of work generated in response to the assessment task. More specifically, the features of quality will be used by teachers, in the process of marking and, where feasible at moderation meetings, to assess and discuss the student's work on the assessment tasks.

Features of Quality: Strand 1 Computer Science Introduction CSI

Achieved with Distinction (90-100%[1])
The main components of a computer system are identified correctly, unambiguously, and communicated clearly. The algorithm is represented correctly and clearly using flow charts and pseudo-code. The algorithm is implemented effectively and creatively; competency in and clear understanding of variables, operators, loops, decisions, and assignment is demonstrated. A very organised and logical approach to the process and stages of testing code is in evidence. Connections are made with team members. Effective and communicative team working is demonstrated.

Achieved with Higher Merit (75-89%)
The main components of a computer system are identified correctly but hesitantly. The algorithm is represented almost correctly and clearly using flow charts and pseudo-code. The algorithm is implemented effectively and creatively; competency in and some understanding of variables, operators, loops, decisions, and assignment is demonstrated. A very organised and logical approach to the process and stages of testing code is in evidence. Connections are made with team members. Effective and communicative team working is demonstrated.

[1] In setting out the Features of Quality, percentage ranges are indicated for each of the grades. These would be used if a marks-based assessment process were being adopted in the assessment of short courses. An alternative is to use a criterion-referenced approach where the descriptions alone are used and the grade rather than a mark is awarded. Feedback on the merits of these two approaches will be sought during the consultation.

Achieved with Merit (55-74%)
Some of the main components of a computer system are identified correctly but hesitantly. The algorithm is represented reasonably clearly using flow charts and pseudo-code. The algorithm is implemented reasonably well using variables, operators, loops, decisions, and assignment. A reasonably organised and logical approach to the process and stages of testing code is in evidence. Connections are made with team members. Effective and communicative team working is in evidence.

Achieved (40-54%)
A few of the main components of a computer system are identified correctly but with uncertainty. The algorithm is represented poorly or incorrectly. The algorithm is implemented using variables, operators, loops, decisions, and assignment. Some capacity for taking a reasonably organised and logical approach to the process and stages of testing code is in evidence. Connections with team members are limited. Team working is ineffective.

Not achieved (0-39%)
Very few of the main components of a computer system are identified correctly. The algorithm is represented incorrectly or not at all. The algorithm is implemented incorrectly with competency using variables, operators, loops, decisions, and assignment not in evidence. The capacity to undertake the process and stages of testing code is not in evidence. Team working is not in evidence.

Features of Quality: Strand 2 Let’s get connected

Achieved with Distinction (90-100%)
There is very clear evidence of an excellent awareness of the mechanics behind the transfer of information across computer networks as well as the reasons behind protocols. The completed web pages show a clear understanding of website construction. The work demonstrates an excellent understanding of how search engines deliver results. There is evidence of an excellent understanding of the representation of different types of data on computers.

Achieved with Higher Merit (75-89%)

There is very clear evidence of a very good awareness of the mechanics behind the transfer of information across computer networks as well as the reasons behind protocols. The completed web pages show a clear understanding of website construction. The work demonstrates a very good understanding of how search engines deliver results. There is evidence of a very good understanding of the representation of different types of data on computers.

Achieved with Merit (55-74%)

There is clear evidence of a good awareness of the mechanics behind the transfer of information across computer networks as well as the reasons behind protocols. The completed web pages show an understanding of website construction. The work demonstrates a good understanding of how search engines deliver results. There is evidence of a good understanding of the representation of different types of data on computers.

Achieved (40-54%)

There is evidence of some awareness of the mechanics behind the transfer of information across computer networks as well as the reasons behind protocols. The completed web pages shows some understanding of website construction. The work demonstrates some understanding of how search engines deliver results. There is evidence of some understanding of the representation of different types of data on computers.

Not achieved (0-39%)

There is little or no evidence of awareness of the mechanics behind the transfer of information across computer networks as well as the reasons behind protocols. The completed web pages shows little or no understanding of website construction. The work demonstrates a limited understanding of how search engines deliver results. There is evidence of little or no understanding of the representation of different types of data on computers.

Features of Quality: Strand 3 Coding at the next level

Achieved with Distinction (90-100%)

There is evidence that the programming tasks are executed with complete confidence and there is a very high level of creativity demonstrated. The tasks demonstrate an excellent understanding and comprehensive knowledge of the advanced concepts of programming and computational thinking. There is evidence that very good connections are made between team members and effective and communicative team working is demonstrated.

Achieved with Higher Merit (75-89%)

There is evidence that the programming tasks are executed with confidence and there is a high level of creativity demonstrated. The tasks demonstrate a very good understanding and comprehensive knowledge of the advanced concepts of programming and computational thinking. There is evidence that good connections are made between team members and effective and communicative team working is demonstrated.

Achieved with Merit (55-74%)

There is evidence that the programming tasks are executed well and there is a good level of creativity demonstrated. The tasks demonstrate a good understanding and knowledge of the advanced concepts of programming and computational thinking. There is evidence that connections are made between team members and effective and communicative team working is demonstrated.

Achieved (40-54%)

There is limited evidence that the programming tasks are executed and some level of creativity demonstrated. The tasks demonstrate a limited understanding and knowledge of the advanced concepts of programming and computational thinking. There is evidence that some connections are made between team members and effective and communicative team working is demonstrated to a limited degree.

Not achieved (0-39%)

There is little or no evidence that the programming tasks are executed and a limited level of creativity demonstrated. The tasks fail to demonstrate an understanding or knowledge of the advanced concepts of programming and computational thinking. There is little or no evidence that some connections are made between team members and effective and communicative team working is demonstrated to little or no degree.

Features of Quality: Strand 4 Problem solving in the real world

Achieved with Distinction (90-100%)

There is evidence of a fully committed consultation process with other members of the group in bringing the final software project through from conception to realisation. There is very comprehensive documentation of the learning process that displays a very clear and logical structure. The project provides excellent examples and proof of a very comprehensive understanding of the concepts in strands 1, 2, and 3. There is very good evidence of a convincing argument put forward to convince others of the benefits of pursuing this challenge further.

Achieved with Higher Merit (75-89%)

There is evidence of a committed consultation process with other members of the group in bringing the final software project through from conception to displays a very clear and logical structure. The project provides very good examples and proof of a comprehensive understanding of the concepts in strands 1, 2, and 3. There is good evidence of a convincing argument put forward to convince others of the benefits of pursuing this challenge further.

Achieved with Merit (55-74%)

There is evidence of a good consultation process with other members of the group in bringing the final software project through from conception to realisation. There is very good documentation of the learning process, which displays a clear and logical structure. The project provides good examples and proof of a very good understanding of the concepts in strands 1, 2, and 3. There is evidence of a convincing argument put forward to convince others of the benefits of pursuing this challenge further.

Achieved (40-54%)

There is limited evidence of a consultation process with other members of the group in bringing the final software project through from conception to realisation. There is some documentation of the learning process, which displays a somewhat unclear and limited logical structure. The project provides some examples and proof of an understanding of the concepts in strands 1, 2, and 3. There is some evidence of an argument put forward to convince others of the benefits of pursuing this challenge further.

Not achieved (0-39%)
There is little or no evidence of a consultation process with other members of the group in bringing the final software project through from conception to realisation. There is limited documentation of the learning process, which displays a somewhat unclear and limited logical structure. The project provides limited examples and some proof of an understanding of the concepts in strands 1, 2, and 3. There is little or no evidence of an argument put forward to convince others of the benefits of pursuing this challenge further.

Resources

This part of the specification will identify resources that will support teaching and learning in the short course.

Appendix

Level Indicators for Level 3 of the National Framework of Qualifications

This short course has been developed in alignment with the Level Indicators for Level 3 of the National Framework of Qualifications. Usually, for Level 3 certification and awards, the knowledge, skill and competence acquired are relevant to personal development, participation in society and community, employment, and access to additional education and training.

NFQ Level	3
Knowledge Breadth	Knowledge moderately broad in range
Knowledge Kind	Mainly concrete in reference and with some comprehension of relationship between knowledge elements
Know-how and skill Range	Demonstrate a limited range of practical and cognitive skills and tools
Know-how and skill Selectivity	Select from a limited range of varied procedures and apply known solutions to a limited range of predictable problems
Competence Context	Act within a limited range of contexts
Competence Role	Act under direction with limited autonomy; function within familiar, homogeneous groups
Competence Learning to Learn	Learn to learn within a managed environment.
Competence Insight	Assume limited responsibility for consistency of self-understanding and behavior.