Національний технічний університет України
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Department of Radio Engineering Systems

OBJECT-ORIENTED PROGRAMMING BASED ON QT TECHNOLOGY

Work program of the academic discipline (Syllabus)

Details of the academic discipline

Level of higher education	**First (bachelor's)**
Field of knowledge	17 — Electronics, Automation, and Electronic Communications
Special	172 — Electronic communications and radio engineering
Educational program	Intelligent technologies of radio-electronic engineering Information and communication radio engineering Radio-technical computerized systems
Status of the discipline	Elective
Form of study	Full-time (day)
Year of study, semester	2nd year, 3rd semester
Scope of the discipline	120 hours (18 hours – lectures, 36 hours – laboratory work, 66 hours – independent study)
Semester control/control measures	Test/test work, Module Control Work
Class schedule	
Language of instruction	Ukrainian
Information about the course supervisor/teachers	Lecturer: Ph.D., Associate Professor Pavlo Yuriyovych Katin, katin.dino@gmail.com, mobile +38(098)202-08-11 Laboratory: Ph.D., Associate Professor Pavlo Yuriyovych Katin,
Course location	

Curriculum

1. Description of the academic discipline, its purpose, subject matter, and outcomes. *The purpose of studying the discipline (credit module) "Object-Oriented Programming Based on Qt Technology" is to provide students with the knowledge and skills to independently develop the architecture and software implementation of prototypes that can form the basis of radio engineering systems and other software solutions.*

C++ syntax in the context of using the language in the Qt library system, software development, debugging, and support technology. Fundamentals of object-oriented programming (OOP) and features of the OOP paradigm in Qt-based programs. Various options for C++-based software solution architectures in the context of using the language in the Qt library system.

Studying this discipline (credit module) allows you to:

create graphical user interfaces using Qt libraries and the C++ programming language;

create programs for working with relational databases using Qt libraries and the C++ programming language;

create programs for working with peripheral device drivers, files, and network technologies;

ensure the platform independence of the resulting software solutions.

2. Prerequisites and post-requisites of the discipline (place in the structural-logical scheme of training under the relevant educational program)

1. *Successful completion of the discipline (credit module) is based on the discipline "Programming" in the curriculum for bachelor's degree programs in specialty 172.*

   The theoretical knowledge and practical skills acquired during the study of the discipline (credit module) are necessary for the development of a bachelor's project.

3. Contents of the discipline

Topic 1. C++ syntax in Qt – basic language constructs, signals and slots, meta-objects.

Topic 2. Development and debugging – Qt Creator, CMake/qmake, profiling, testing. Topic 3. OOP in Qt –

encapsulation, inheritance, polymorphism, QObject, and QWidgets.

Topic 4. Architectures – MVC, MVVM, design patterns, signal-slot model. Topic 5. GUI – Qt Widgets, Qt

Quick, QML, interface creation.

Topic 6. Databases – Qt SQL, SQLite/MySQL/PostgreSQL, CRUD operations.

Topic 7. Drivers, files, networks – QFile, QIODevice, TCP/UDP client-server.

Topic 8. Platform dependency – Qt cross-platform compatibility, conditional compilation, testing on different operating systems.

Topic 9. Final review – knowledge integration, development of a comprehensive application. Modular

test

4. Teaching materials and resources

Main literature:

1. Stroustrup, B. The C++ Programming Language. 4th ed. – Boston: Addison-Wesley, 2013. – 1376 p.

2. Blanchette, J., Summerfield, M. C++ GUI Programming with Qt 4. – Upper Saddle River: Prentice Hall, 2006. – 512 p.

3. Summerfield, M. Advanced Qt Programming: Creating Great Software with C++ and Qt 4. – Upper Saddle River: Addison-Wesley, 2010. – 528 p.

Additional reading:

1. Blanchette, J., Summerfield, M. C++ GUI Programming with Qt 5. – Upper Saddle River : Prentice Hall, 2016. – 624 p.

2. Meyers, S. Effective C++: 55 Specific Ways to Improve Your Programs and Designs. 3rd ed. – Boston : Addison-Wesley, 2005. – 320 p.

3. Qt Documentation [Electronic resource]. – Access mode: <https://doc.qt.io>. – Date of access: 01.12.20

Educational content

1. Methodology for mastering the academic discipline (educational component).

No	Type of classes	Description of class content
Topic 1. C++ syntax in Qt – basic language constructs, signals and slots, meta-objects.		
1	Lecture 1. **Fundamentals of of C++ syntax and integration with Qt**	Key C++ constructs (classes, templates, exceptions) are discussed. The role of the preprocessor, namespaces, and the standard library are explained. The mechanism of signals and slots in Qt as an extension of C++ is demonstrated. An example of a simple Qt program using basic elements is shown.
2		
3	Laboratory work 1.	Introduction to C++ syntax and creation of your first Qt program
Topic 2. Development and debugging – Qt Creator, CMake/qmake, profiling, testing.		
4	Lecture 2. Development and debugging tools in Qt Creator	Overview of the Qt Creator environment, qmake and CMake build systems. Methods of debugging programs: breakpoints, viewing variables, profiling. Using unit tests to verify functionality. Practical aspects of code maintenance and documentation.
5		
6	Lab work 2.	Using signals and slots in Qt
Topic 3. OOP in Qt – encapsulation, inheritance, polymorphism, QObject, and QWidgets.		
7	Lecture 3. Object-oriented programming and the Qt paradigm paradigm	Encapsulation, inheritance, and polymorphism in C++. Features of the QObject class and its role in Qt. Using virtual methods and function overriding. Examples of creating custom widgets based on Qt base classes.
8		
9	Laboratory work 3.	Implementation of classes and inheritance in Qt programs
Topic 4. Architectures – MVC, MVVM, design patterns, signal-slot model.		
10	Lecture 4. Architectural approaches in C++/Qt	Overview of MVC and MVVM models in the context of Qt. Signal-slot architecture as the basis for component interaction. Use of design patterns , Observer, Singleton, Factory).
11		

		Examples of building small systems using architectural solutions.
12	*Laboratory work 4.*	*Building application architecture (MVC/MVVM)*
Midterm Test		
Topic 5. GUI – Qt Widgets, Qt Quick, QML, interface creation.		
13	*Lecture 5. Creation Graphical interfaces*	*Basics of working with Qt Widgets: buttons, menus, dialogs. Using Qt Designer to quickly create interfaces. Introduction to QML and its role in modern UIs. Examples of combining QML and C++ in a single application.*
14		
15	*Lab work 5.*	*Creating a graphical user interface with Qt Widgets*
Topic 6. Databases – Qt SQL, SQLite/MySQL/PostgreSQL, CRUD operations.		
16	*Lecture 6. Qt SQL and integration with databases*	*Overview of the Qt SQL Module and its capabilities. Connecting to SQLite, MySQL, and PostgreSQL. Performing CRUD operations via Qt. Examples of creating an application with a database.*
17		
18	*Lab 6.*	*Working with databases in Qt SQL (CRUD operations)*
Topic 7. Drivers, files, networks – QFile, QIODevice, TCP/UDP client-server.		
19	*Lecture 7. Qt Network and file operations*	*Using QFile and QIODevice classes to work with files. Basics of working with TCP and UDP network protocols. Creating client-server applications in Qt. Examples of interaction with peripheral devices.*
20		
21	*Laboratory work 7.*	*Working with files and network technologies in Qt*
Topic 8. Platform dependency – Qt cross-platform compatibility, conditional compilation, testing on different operating systems.		
22	*Lecture 8. Cross-platform compatibility and conditional compilation.*	*Features of Qt on Windows, Linux, and macOS. Using conditional compilation (#ifdef, #endif). Testing programs on different operating systems. Practical examples of adapting code for different platforms.*
23		
24	*Lab work 8.*	*Cross-platform testing and adaptation of software*
Topic 9. Final review – integration of knowledge, development of a comprehensive application.		

25	Lecture 9. Comprehensive application development in C++/Qt	Generalization of knowledge of previous modules. Designing a comprehensive application with a GUI, database and network capabilities. Discussion of typical problems and ways to solutions. Preparation for defending course project.
26		
Credit		

6. Independent work of a student/graduate student

The discipline is based on independent preparation for classroom sessions on theoretical and practical topics.

No .	Area of independent preparation	Number of hours	Literature
1	Preparation for Lecture 1	1	1-5
2	Preparation for laboratory work 1	1.5	1
3	Preparation for lecture 1	1	1
4	Preparation for lecture 2	1	1
5	Preparation for laboratory work 2	1.5	1
6	Preparation for lecture 2	1	1
7	Preparation for lecture 3	1	1
8	Preparation for laboratory work 3 (part 1)	1.5	1
9	Preparation for lecture 3	1	1
10	Preparation for lecture 4	1	1
11	Preparation for lab work 3 (part 2)	1.5	1-5
12	Preparation for lecture 4	1	1
13	Preparation for lecture 5	1	1
14	Preparation for laboratory work 4 (part 1)	1.5	1
1	Preparation for lecture 5	1	1
16	Preparation for lecture 6	1	1
17	Preparation for laboratory work 4 (part 2)	1.5	
1	Preparation for lecture 7	1	
19	Preparation for lecture 7	1	
20	Preparation for laboratory work 5	1.5	
21	Preparation for lecture 8	1	1
2	Preparation for lecture 8	1	1
23	Preparation for Laboratory Work 6 (part 1)	1.5	1
2	Preparation for lecture 16	1	1

25	*Preparation for lecture 9*	*1*	*1*
26	*Preparation for laboratory work 6 (part 2)*	*1.5*	*1*
27	*Preparation for the midterm test*	*4*	*1*
28	*Qt SQL Module*	*30*	*1-5*
29	*MVC and MVVM in the context of Qt.*	*2*	*1*
30	*SQLite, MySQL, and PostgreSQL*	*1*	*1*
31	*Essential C++ Features for web developing. Stage 1.*	*1*	*1*
32	*Essential C++ Features for web developing. Stage 1.*	*1*	*1*
33	*Essential C++ Features for web developing. Stage 2.*	*1.5*	*1*
34	*Essential C++ Features for web developing. Stage 2.*	*1*	*1*
35	*Essential C++ Features for web developing. Stage 3.*	*1*	*1*
36	*Essential C++ Features for web developing. Stage 3.*	*2*	*1*
37	*Essential C++ Features for web developing. Stage 4.*	*2*	*1*
38	*Essential C++ Features for Web Development. Stage 4.*	*1*	*1*
39	*QFile and QIODevice*	*1*	*1*
40	*TCP and UDP.*	*2*	*1*

Policy and control

7. Academic discipline policy (educational component)

Attendance at lectures is mandatory. In exceptional circumstances, attendance requirements and other aspects of the policy may be subject to change.

Attendance at computer lab classes may be sporadic and, if necessary, for consultation/defense of laboratory work.

Rules of conduct in class: be active, respect others, turn off phones. Follow the academic integrity policy.

Rules for defending laboratory work: work must be done in accordance with the tasks set and according to the option.

8. Types of control and rating system for assessing learning outcomes (RSO)

During the semester, students complete 8 laboratory works. The maximum number of points for each: 10 points.

Points are awarded for:
- quality of laboratory work: 0-5 points;
- answers during the defense of laboratory work: 0-3 points;
- timely submission of work for defense: 0-2 points.

Criteria for assessing the quality of performance:
5 points – work performed well, in full;
4 points – the work is done well, in full, but has some flaws;
3 points – the work is completed in full, but contains minor errors; 2 points – the work is completed in full, but contains significant errors; 0 points – the work is not completed in full.

Criteria for evaluating answers:
3 points – the answer is complete and well-reasoned;

2 points – the answer is correct, but has shortcomings or minor errors; 1 point – the answer contains significant errors;
0 points – no answer or incorrect answer.

Criteria for assessing the timeliness of the submission of work for defense: 2 points – the work is submitted for defense no later than the specified deadline;
0 points – the work is submitted for defense after the specified deadline.

Maximum number of points for completing and defending laboratory work: 10 points × 8 lab sessions = 80 points.

During the semester, quizzes on the topic of the current lesson are held during lectures. Maximum number of points for all quizzes: 3 points. The number of quizzes on the topic of the current lesson for one student is unlimited.

The module test consists of 1 theoretical and 1 practical question. The answer is graded on a scale of up to 20 points.

The criteria for evaluating each question of the test are as follows:
9-10 points – the answer is correct, complete, and well-reasoned;
7-8 points – the answer is correct, detailed, but not very well argued;
5-6 points – the answer is generally correct, but has some flaws;
3-4 points – the answer contains minor errors;
1-2 points – the answer contains significant errors;
0 points – no answer or incorrect answer.

Maximum number of points for the module test: 10 points × 2 questions = 20 points.

The rating scale for the discipline is:
$R = _{RC}$ = 80 points + 20 points = 100 points.

Calendar control: conducted twice per semester as monitoring of the current status of syllabus requirements fulfillment.
At the first assessment (week 8), the student receives a "pass" if their current rating is at least 15 points (50% of the maximum number of points a student can receive before the first assessment).
At the second assessment (week 14), students receive a "pass" if their current rating is at least 20 points (50% of the maximum number of points a student can receive before the second assessment).

Semester control: credit
Conditions for admission to semester control:
With a semester rating ($_{RC}$) of at least 60 points and all computer workshop assignments completed, the student automatically receives a credit according to the table (Table of correspondence of rating points to grades on the university scale). Otherwise, they must complete a credit test.

A prerequisite for admission to the credit test is the completion and defense of laboratory work.
If a student does not agree with the automatic grade, they can try to improve their grade by writing a credit test, in which case their points earned during the semester are retained, and the better of the two grades received by the student is awarded (a "soft" grading system).

Table of correspondence between rating points and grades on the university scale:

Number of points	*Rating*
100-95	Excellent
94	Very good
84	Good
74-65	Satisfactory
64-60	Sufficient
Less than 60	Unsatisfactory
Admission requirements not met	Not admitted

9. Additional information on the discipline (educational component)

Various operating systems may be used for work at the student's discretion and in agreement with the instructor.

If the student wishes to use other programming languages or technologies in the computer lab, they can be included as an additional element in the basic architecture of the distributed web application.

Work program of the academic discipline (syllabus):

Compiled by *Associate Professor of the Department, Ph.D., Associate Professor, Katin P.Yu.*

Approved by *the RTS Department (Minutes No. 06/2024_ dated 27.06.2024)*

Approved by *the Methodological Commission of the Radio Engineering Faculty (Minutes No. 06/2024 dated June 28, 2024)*