



**HW (LC)****HW (PD)****MB****SW****SSW**

1.1 Основи організації та функціонування комп'ютерів

Основи алгоритмізації та програмування (П.ч.1)

1.2 Комп'ютерна логіка, частина 1

Дискретна математика

Об'єктно-орієнтоване Програмування (П.ч.2)

2.1 Комп'ютерна логіка, частина 2

Теорія електричних та магнітних кіл  
Комп'ютерна електроніка

Структури даних та алгоритми (П.ч.3)

2.2 Комп'ютерна схемотехніка

Алгоритми та методи обчислень

Засоби системного програмування

3.1 Архітектура комп'ютерів

Системне програмування

3.2 Комп'ютерні системи

Паралельні та розподілені обчислення

Системне програмне забезпечення

4.1

4.2 Мікропроцесорні системи

Комп'ютерні мережі

Обробка сигналів


**School of Engineering**

Aeronautics and  
Astronautics Fields (PhD)

Aerospace Engineering  
(Course 16)

Archaeology and Materials  
(Course 3-C)

Artificial Intelligence and  
Decision Making  
(Course 6-4)

Biological Engineering  
(Course 20)

Biological Engineering (PhD)

Chemical-Biological  
Engineering (Course 10-B)

Chemical Engineering  
(Course 10)

Chemical Engineering  
(Course 10-C)

Computer Science and  
Engineering (Course 6-3)

Electrical Engineering and  
Computer Science  
(Course 6-2)

Electrical Engineering and  
Computer Science (MEng)

Electrical Science and  
Engineering (Course 6-1)

Engineering (Course 1-FNG)

The General Institute Requirements include a Communication Requirement that is integrated into both the HASS Requirement and the requirements of each major; see details below.

**Summary of Subject Requirements**
**Subjects**

Science Requirement	6
---------------------	---

Humanities, Arts, and Social Sciences (HASS) Requirement; at least two of these subjects must be designated as communication-intensive (CI-H) to fulfill the Communication Requirement.	8
---	---

Restricted Electives in Science and Technology (REST) Requirement [satisfied by 18.C06[J] and 6.1910, 6.2000, 6.3700, or 18.05 in the Departmental Program]	2
---	---

Laboratory Requirement (12 units) [can be satisfied by 6.3100 in the Departmental Program]	1
--	---

<b>Total GIR Subjects Required for SB Degree</b>	<b>17</b>
--	-----------

**Physical Education Requirement**

Swimming requirement, plus four physical education courses for eight points.

**Departmental Program**

Choose at least two subjects in the major that are designated as communication-intensive (CI-M) to fulfill the Communication Requirement.

---

Students must satisfy at least one program requirement or elective with a subject from the Project-Based Laboratory (PLAB) list<sup>1</sup>

**Fundamentals**

6.100A	Introduction to Computer Science Programming in Python	6-9
--------	--	-----

or 6.100L	Introduction to Computer Science and Programming	
-----------	--	--

6.120A	Discrete Mathematics and Proof for Computer Science	6-12
--------	---	------

or 6.1200[J]	Mathematics for Computer Science	
--------------	----------------------------------	--

6.1210	Introduction to Algorithms	12
--------	----------------------------	----



<https://www.acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf>

acm.org/binaries/content/assets/education/curricula-recommendations/cc2020.pdf

27 / 205 | - 100% + | ☰ ⚡

08April2021-Report

27

28

29

30

### 2.3.1: Computer Engineering

Computer engineering (CE) brings together computing and electrical engineering in a way that embodies the science and technology of design, construction, implementation, and maintenance of software and hardware components of modern computing systems, computer-controlled equipment, and networks of intelligent devices. CE is the computing discipline that explicitly focuses the development of hardware and software interface as a hardware embedded element of a computing system. The *Computer Engineering Curricula 2016 Report*, known also as CE2016, represents curriculum guidelines for undergraduate degree programs in computer engineering [Acm06]. The goals of the effort include incorporating past and future development needs, supporting professionals responsible for teaching a range of degree programs in computer engineering worldwide.

The capabilities of CE graduates integrate aptitudes of electrical engineering, software engineering, and computer science with a heavy emphasis on mathematics required as a foundation. CE2016 is noticeably clear about the fact that graduates from CE programs should have the ability to design computers, design computer-based systems, and design networks with additional specifications that design needs to exceed simple configuration and assembly. CE is specifically an engineering discipline where graduates must have a breadth of knowledge in mathematics and engineering sciences with a preparation for professional practice or graduate work in engineering. Many countries provide CE graduates the opportunity to become licensed professional engineers according to local governmental rules.

The computer engineering discipline enables graduates to analyze and design circuits, manage the design of computer hardware components, and develop networking hardware solutions. For students interested in gaining experiences in integrating computing capabilities directly with computing hardware, computer engineering could be an appropriate degree program choice. Computer engineering also provides an excellent preparation for the design and development of modern technologies that tightly integrate the physical world with the world of the artificial.

### 2.3.2: Computer Science

The *Computer Science Curricula 2013* project had two directives in developing its subsequent report, known as CS2013 [Acm04]. They included (1) a review of Computing Curriculum 2001 and CS2008, and (2) seeking input from diverse audiences to broaden participation in computer science (CS). CS2013 also had several high-level themes that provided overarching guidance for the development of its report. These include embracing an outward-looking view of the discipline, size management of the curriculum, providing actual exemplars to identify and describe existing successful courses and curricula, and being responsive to institutional needs, goals, and resource constraints.

Because of its theoretical foundations, computer science is often viewed as a fundamental discipline. It is, however, at times erroneously equated with all of computing. This misconception is understandable given that the theoretical roots of computer science have emerged separately from the engineering tradition of computing's earliest days [Fed1 p3.2]. While the physical sciences are fundamental and offer theoretical basis to engineering fields, none subsumes the other and each has a well understood distinct identity. Similarly, this Report and its predecessors have successfully established independent identities relative to computer science.

CS continues to have a more theoretical focus among the other computing disciplines, and its connection with abstract mathematics is still strong. A CS degree alone typically does not provide expertise regarding a specific context applicable to computing. Instead, CS programs emphasize abstract computational capabilities. CS2013 identifies

	<u>HW (LC)</u>	<u>HW (PD)</u>	<u>MB</u>	<u>SW</u>	<u>SSW</u>
1.1	Основи організації та функціонування комп'ютерів			Основи алгоритмізації та програмування (П.ч.1)	
1.2	Комп'ютерна логіка, частина 1		Дискретна математика	Об'єктно-орієнтоване Програмування (П.ч.2)	
2.1	Комп'ютерна логіка, частина 2	Теорія електрических та магнітних кіл Комп'ютерна електроніка		Структури даних та алгоритми (П.ч.3)	
2.2	Комп'ютерна схемотехніка		Алгоритми та методи обчислень		Засоби системного програмування
3.1	Архітектура комп'ютерів				Системне програмування
3.2	Комп'ютерні системи		Паралельні та розподілені обчислення		Системне програмне забезпечення
4.1					
4.2	Мікропроцесорні системи		Комп'ютерні мережі	Обробка сигналів	

# Senior Research Scientist, Circuits

[https://nvidia.wd5.myworkdayjobs.com/en-US/NVIDIAExternalCareerSite/job/Taiwan-Taipei/Senior-Research-Scientist--Circuits\\_JR1965230?jobFamilyGroup=0c40f6bd1d8f10ae43ffc8817cf47e8e](https://nvidia.wd5.myworkdayjobs.com/en-US/NVIDIAExternalCareerSite/job/Taiwan-Taipei/Senior-Research-Scientist--Circuits_JR1965230?jobFamilyGroup=0c40f6bd1d8f10ae43ffc8817cf47e8e)

# Senior Physical Design Methodology Engineer

[https://nvidia.wd5.myworkdayjobs.com/en-US/NVIDIAExternalCareerSite/job/Senior-Physical-Design-Methodology-Engineer\\_JR1994272](https://nvidia.wd5.myworkdayjobs.com/en-US/NVIDIAExternalCareerSite/job/Senior-Physical-Design-Methodology-Engineer_JR1994272)

# Високорівневе проектування комп'ютерних систем і автоматичний синтез до нижчих рівнів.



OPEN SYSTEMC INITIATIVE  
Defining & Advancing SystemC Standards



Overview

About SystemC

Regional User Groups

Resources

[Home](#) » [SystemC Community](#) » **About SystemC**

## About SystemC

### **The Language for System-Level Modeling, Design and Verification**

Ratified as IEEE Std. 1666™-2005, SystemC™ is a language built in standard C++ by extending the language with the use of class libraries. SystemC addresses the need for a system design and verification language that spans hardware and software. The language is particularly suited to model system's partitioning, to evaluate and verify the assignment of blocks to either hardware or software implementations, and to architect and measure the interactions between and among functional blocks. Leading companies in the intellectual property (IP), electronic design automation (EDA), semiconductor, electronic systems, and embedded software industries currently use SystemC for architectural exploration, to deliver high-performance hardware blocks at various levels of abstraction and to develop virtual platforms for hardware/software co-design.

# Високорівневе проектування комп'ютерних систем і автоматичний синтез до нижчих рівнів.

Standard

Active

## IEEE 1800-2017 - IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language

ACCESS VIA THE IEEE GET PROGRAM

ACCESS VIA SUBSCRIPTION

### Explore This Standard



Standard Details



Additional Resources



Working Group

### Standard Details

The definition of the language syntax and semantics for SystemVerilog, which is a unified hardware design, specification, and verification language, is provided. This standard includes support for modeling hardware at the behavioral, register transfer level (RTL), and gate-level abstraction levels, and for writing testbenches using coverage, assertions, object-oriented programming, and constrained random verification. The standard also provides application programming interfaces (APIs) to foreign programming languages. (The PDF of this standard is available at no cost at <https://ieeexplore.ieee.org/browse/standards/get-program/page> compliments of Accellera Systems Initiative)

Sponsor  
Committee

C/DA - Design Automation

# Високорівневе проектування комп'ютерних систем і автоматичний синтез до нищих рівнів.

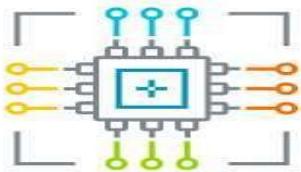


The image shows the homepage of the Chisel/FIRRTL Hardware Compiler Framework. The background is a blue gradient with a repeating pattern of the word "CHISEL" in white. At the top left is a logo with a stylized "C" and "H" inside a circle, next to the text "Chisel/FIRRTL". To the right is a search bar with a magnifying glass icon and the placeholder "Enter keywords here...". Further right are links for "GitHub" and "Documentation". The main title "Chisel/FIRRTL Hardware Compiler Framework" is centered in large white font. Below it is a white button with the text "View on GitHub". At the bottom of the page, there is a horizontal navigation bar with links: "Chisel3", "Testers", "ChiselTest", "FIRRTL", "Treadle", "Diagrammer", and "Community".

CHISEL



# ARM



## Processor IP >

Arm is the leading technology provider of processor IP, offering the widest range of processors to address the performance, power, and cost requirements of every device. Arm CPUs and NPUs include Cortex-A, Cortex-M, Cortex-R, Neoverse, Ethos and SecurCore.



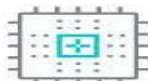
## Security IP >

CryptoCell, TrustZone, SecurCore, Cortex-M35P



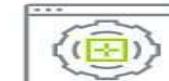
## System IP >

CoreLink, CoreSight, Coherent Mesh Network, AMBA and more



## Graphics and Multimedia >

Arm Mali GPUs and Mali Camera series of ISPs



## Software and Development Tools >

Keil RTX5, Allinea Studio, Compilers, Debuggers and more

### All IP Products

#### Processors

#### Graphics and multimedia processors

#### System IP

#### Physical IP

#### Security IP

#### Subsystem

### All IP Products

#### Arm DevSummit

Join our global community of hardware designers and software developers for an online event that puts cutting-edge technology right at your fingertips.

#### Processors

Design, verify and program Arm processors.

#### Graphics and Multimedia Processors

Arm Mali multimedia IP includes Mali graphics processors and Mali camera image signal processors.

#### System IP

Configure and build performant, power efficient SoCs, which you can differentiate by combining Arm processors with your own IP elements using AMBA interfaces.

#### Physical IP

Artisan Physical IP delivers the most comprehensive and advanced physical IP solution in the industry. Available for free through the DesignStart Tier of Arm Flexible Access, Artisan Physical IP is the fastest, simplest route to a custom System on Chip.

#### Security IP

Explore subsystems that provide platform level security, acceleration and offloading.

#### Subsystem

Learn more about Corstone, and directly integrate solutions into your design.

### All Tools and Software

### IP configuration tools

### Open Source Software

### Embedded

### Server and HPC

### Graphics and gaming

### Development boards

### Simulation models

### Licensing

### Simulation models

#### Arm Flexible Access Models

The Arm Flexible Access Model package allows SoC architects to make a well-informed decision on what Arm IP and IP configuration to choose for their next project.

#### DesignStart simulation models

DesignStart simulation models allow developers to evaluate the performance of Arm technology, and enable software teams to begin sooner.

#### Fast Models

Fast Models are accurate, flexible programmer's view models of Arm IP, allowing you to develop software like drivers, firmware, OS, and applications before silicon availability.

### Cycle Models

Cycle Models are compiled directly from Arm RTL and retain complete functional and cycle accuracy. This enables you to confidently make architectural decisions, optimize performance, or develop bare-metal software.

### Fixed Virtual Platforms

Running at speeds comparable to the real hardware, Fixed Virtual Platforms are complete simulations of an Arm system, including processor, memory, and peripherals.

### AMBA TLM library

The Arm AMBA Transaction-Level Modeling (TLM) library allows you to model and simulate approximately-timed and cycle-accurate AXI4 and ACE ports.

# Cycle Models

Home   Models   Platforms   Support

## Cortex A-Series

- Cortex-A32
- Cortex-A35
- Cortex-A53
- **Cortex-A55**

## Cortex R-Series

- Cortex-R5
- Cortex-R52
- Cortex-R8
- Cortex-R82
- Cortex-R52+

## Cortex M-Series

- Cortex-M0+
- Cortex-M23
- Cortex-M33
- Cortex-M7
- Cortex-M55

## Cortex-A55 Cycle Model

The Arm® Cortex-A55 core delivers the best combination of power efficiency and performance in its class. It is part of the first generation of application CPUs based on DynamIQ technology and features the latest Armv8-A architecture extensions, with dedicated machine learning instructions. The Cortex-A55 Cycle Model includes the DynamIQ Shared Unit (DSU).

DSU r4p1 supports Cortex-A55 r2p0.

**Supported Rxpy Rev:** r4p1

**Supported Platforms:** SystemC, PlatformArchitect, RSM

## AMBA TLM Library

[Overview](#) [Arm Flexible Access Models](#) [DesignStart Simulation Models](#) ▾ [Fast Models](#) ▾ [Cycle Models](#) ▾ [Fixed Virtual Platforms](#) ▾ [AMBA TLM](#) ▾

## Overview

The Arm AMBA Transaction-Level Modeling (TLM) library allows you to model and simulate approximately-timed (AT) and cycle-accurate (CA) AXI4 and ACE ports. This C++ library is provided as a pre-compiled binary library.

This library is intended for use with:

- Arm Cycle Models
- Custom AT and CA models used for virtual prototype development
- Third-party models that use supported AMBA interfaces

# RISC-V



About RISC-V ▾ Membership ▾ RISC-V Exchange ▾ Technical ▾ News & Events ▾ Community ▾ Q

## RISC-V Exchange: Cores & SoCs

This page is a collection of available intellectual property (IP) cores and SoCs in the RISC-V ecosystem. This list is curated by the community – which includes you! Add cores, SoC platforms, and SoCs to the list by filing a pull request on the [GitHub repository](#). If you have any questions about this process, [contact us](#) for help.

Please note that the Exchange can showcase available physical hardware on the [Available Boards](#) page.

[Cores](#)   [SoC Platforms](#)   [SoCs](#)

Search:

Name	Supplier	Links	Capability	Priv. spec	User spec	Primary Language	License
RV32EC_P2	IQonIC Works	<a href="#">Website</a>	RV32	1.11	RV32E[M]C/RV32I[M]C	SystemVerilog	IQonIC Works Commercial License
RV32IC_P5	IQonIC Works	<a href="#">Website</a>	RV32	1.11	RV32I[M][N][A]C	SystemVerilog	IQonIC Works Commercial License



# Wishbone

The screenshot shows a web browser displaying the OpenCores website at [opencores.org/howto/wishbone](http://opencores.org/howto/wishbone). The page title is "SoC Interconnection: WISHBONE".

**Login Form:**

Username: \_\_\_\_\_  
Password: \_\_\_\_\_  
 Remember me  
**Buttons:** Login, Register

**Browse Projects:**

- PROJECTS
- FORUMS
- ABOUT
- HowTo/FAQ
- MEDIA
- LICENSING
- PARTNERS
- MAINTAINERS
- CONTACT US

**Description:**

The WISHBONE System-on-Chip (SoC) Interconnect Architecture for Portable IP Cores is a portable interface for use with semiconductor IP cores. Its purpose is to foster design reuse by alleviating system-on-a-chip integration problems. This is accomplished by creating a common, logical interface between IP cores. This improves the portability and reliability of the system, and results in faster time-to-market for the end user. WISHBONE itself is not an IP core...It is a specification for creating IP cores.

OpenCores recommends the WISHBONE System-on-Chip Interconnect as the interface to all cores that require interfacing to other cores inside a chip (FPGA, ASIC, etc.).

The WISHBONE standard is not copyrighted, and is in the public domain. It may be freely copied and distributed by any means. Furthermore, it may be used for the design and production of integrated circuit components without royalties or other financial obligations.

**News:**

- 2010-06-22, OpenCores Releases WISHBONE Rev.B4 specs.
- 2002-07-09, OpenCores Releases WISHBONE Rev.B3 specs.

**Additional information:**

Download a copy of the [WISHBONE, Revision B.4 Specification](#) - Adobe Acrobat ".pdf", 1.1 MiB. This new revision B4 supports pipeline traffic mode.

Download a copy of the [WISHBONE, Revision B.3 Specification](#) - Adobe Acrobat ".pdf", 900 Kib.

**Comparison to other SoC buses:**

2001-01-09 "Review of Three SoC Buses", Rudolf Usselmann [soc\\_bus\\_comparison.pdf](#) (79 Kb)

**Application notes:**

2001-04-18 "Combining WISHBONE interface signals", Richard Hervelle [appnote\\_01.pdf](#) (21 Kb)

# Wishbone

FPGA Ethernet Cores

Connect your FPGA.

Subscribe to news

Your Name (required)

Your Email (required)

SUBSCRIBE

Follow @FPGA\_Cores

WISHBONE

Wishbone is an open source standard bus that connects slave peripherals to a master CPU. Instant SoC V1.2 supports Wishbone and you can easily add your own VHDL or Verilog peripherals to the Instant SoC RISC-V system. Instant SoC supports the B4 version of Wishbone.

The diagram illustrates the Wishbone Bus interface. On the left, a blue box labeled "Instant SoC Wishbone Master" contains pins: SysCik, RST\_O, ADR\_O(), DAT\_I(), DAT\_O(), WE\_O, STB\_O, ACK\_I, and CYC\_O. On the right, a blue box labeled "Wishbone Slave" contains pins: RST\_I, CLK\_I, ADR\_I(), DAT\_I(), DAT\_O(), WE\_I, STB\_I, ACK\_O, and CYC\_I. Arrows show the connections from the Master's ADR\_O() to the Slave's ADR\_I(), DAT\_O() to DAT\_I(), WE\_O to WE\_I, STB\_O to STB\_I, ACK\_I to ACK\_O, and CYC\_O to CYC\_I. The Master's RST\_O is connected to the Slave's RST\_I. The Master's SysCik is also connected to the Slave's CLK\_I.

Wishbone Bus

To add a Wishbone component you simply creates a [FC\\_Wishbone](#) object in C++ with a file path to the component and Instant SoC adds the component and maps all Wishbone bus signals to the system. The file could be a Verilog (\*.v) or VHDL (\*.vhd) source file. You can add any number of the component to the `ecustom`

# CoreConnect

**IBM.**

Country/region [select] | Terms of use  
Search

Home | Products | Services & solutions | Support & downloads | My account

**Semiconductor solutions**

- ASIC
- Foundry
- Power Architecture
- Design Centers
- Manufacturing
- Design Libraries and Intellectual Property
- Packaging
- Services
- Ready for IBM Technology
- Contact us

**Documentation**

**News & events**

**Related links**

- IBM Customer Connect
- Global Engineering Solutions

## CoreConnect Bus Architecture

Related links: PowerPC, Technical Library

**Categories**

[CoreConnect PLB4 Bus Cores](#)  
[CoreConnect PLB4 Peripheral Cores](#)

(Click on column header to sort)

Documents	Type	Date (mm/dd/yy)
<a href="#">Processor Local Bus (128-bit)</a>	Specifications	05/02/07
<a href="#">IBM CoreConnect and CPU support cores</a>	Product Brief	07/24/06
<a href="#">IBM CoreConnect Bus Cores</a>	Product Brief	07/24/06
<a href="#">Device Control Register Bus 3.5 Architecture Specifications</a>	Specifications	01/27/06
<a href="#">CoreConnect FAQ</a>	FAQ	09/06/02
<a href="#">DCR Addressing with the CoreConnect DCR Master</a>	Application Note	05/08/02
<a href="#">On-Chip Peripheral Bus</a>	Specifications	04/01/01
<a href="#">CoreConnect Bus Architecture</a>	Product Brief	09/01/99
<a href="#">CoreConnect Bus Architecture</a>	White Paper	09/01/99
<a href="#">IBM On-Chip Bus Model Toolkits</a>	Product Brief	06/01/98

**IBM Customer Connect**  
→ Sign in

**IBM microNews**  
→ Latest news from IBM Technology and Power Architecture

**Feedback**  
→ Questions or comments on the technical library

**Help**  
□ Information on search and navigation

# AMBA



Products



Solutions



Why Arm



Support &

Training



Resources



Company

arm

SYSTEM IP

**AMBA**

## AMBA: The Standard for On-Chip Communication

AMBA is a freely available open standard for the connection and management of functional blocks in a system-on-chip. AMBA specifications are widely adopted as the standard for on-chip communication and provide a standard interface for IP re-use. This helps reduce the risks and costs of developing multi-processor designs with many controllers and peripherals. Different AMBA specifications outline the interfaces and protocols for use in applications across multiple market areas. Two key specifications include CHI and AXI. CHI defines the architecture for fully coherent, high-performance multi-core systems; AXI is used for a wide range of high-performance applications, including mobile computing, networking, automotive, and high-performance IoT.

# Проектування комп'ютерних систем на рівні топології кристалу



Industry Solutions ▾

Software & Products ▾

Solutions & Services ▾

Training & Support ▾



Sign in

Home > Siemens EDA Software

## Electronic Design Automation

Siemens EDA: Where electronic innovation meets tomorrow.

[Read Our Latest Blogs](#)



## Siemens EDA

The pace of innovation in electronics is constantly accelerating. To enable our customers to deliver life-changing innovations to the world faster and to become market leaders, we are committed to delivering the world's most comprehensive portfolio of electronic design automation (EDA) software, hardware, and services.

[White Paper: Engineer a Smarter Future Faster](#)



# Проектування комп'ютерних систем на рівні топології кристалу



SILICON DESIGN & VERIFICATION

SILICON IP

SOFTWARE INTEGRITY

ABOUT US

Support ▾

Global Sites ▾



## Leading the Era of Autonomous Chip Design

Achieve PPA Targets Faster with DSO.ai

Play Video

Discover How

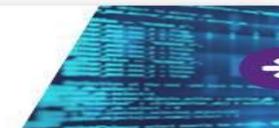
Silicon Design & Verification



Silicon IP



Software Security & Quality



# Проектування комп'ютерних систем на рівні топології кристалу

[Products](#)[Solutions](#)[Support](#)[Company](#)

EN

US

## DESIGN EXCELLENCE

[Digital Design and Signoff](#)[Custom IC](#)[Verification](#)[IP](#)[IC Package](#)

## SYSTEM INNOVATION

[System Analysis](#)[Embedded Software](#)[PCB Design](#)

## PERVASIVE INTELLIGENCE

[AI / Machine Learning](#)[AI IP Portfolio](#)

## CADENCE CLOUD

[VIEW ALL PRODUCTS](#)

## Digital Design and Signoff

Cadence® digital design and signoff solutions provide a fast path to design closure and better predictability, helping you meet your power, performance, and area (PPA) targets.

### PRODUCT CATEGORIES

- [Logic Equivalence Checking](#)
- [SoC Implementation and Floorplanning](#)
- [Functional ECO](#)
- [Low-Power Validation](#)
- [Synthesis](#)
- [Power Analysis](#)
- [Constraints and CDC Signoff](#)
- [Silicon Signoff and Verification](#)
- [Library Characterization](#)
- [Test](#)

### FEATURED PRODUCTS

- [Cerebrus Intelligent Chip Explorer](#)
- [Genus Synthesis Solution](#)
- [Conformal Smart LEC](#)
- [Innovus Implementation System](#)
- [Tempus Timing Signoff Solution](#)
- [Voltus IC Power Integrity Solution](#)
- [Pegasus Verification System](#)

### RESOURCES

- [Flows](#)

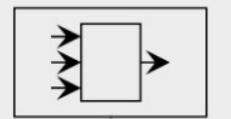


# Проектування комп'ютерних систем на рівні топології кристалу

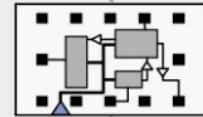
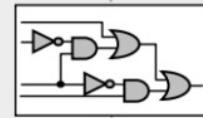
- Analog Design Environment:**
  - Cadence Spectre
  - ADE L/XL/GXL
- Layout Design**
  - Virtuoso Layout Suite L/XL/GXL
- DFT**
  - TetraMax for ATPG
- Synthesis & Lint**
  - Spyglass
- STA**
  - Primetime-SI
- Formal Verification**
  - Conformal

- Place & Route**
  - Innovus
- Extraction**
  - Quantus
- Physical Verification**
  - Calibre LVS/DRC
- Power & IR Drop Analysis**
  - Voltus
- Design Verification**
  - Incisive

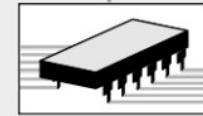
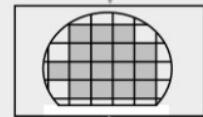
# VLSI Design Flow



ENTITY test is  
port a: in bit;  
end ENTITY test;



DRC  
LVS  
ERC



## System Specification

## Architectural Design

## Functional Design and Logic Design

## Circuit Design

## Physical Design

## Physical Verification and Signoff

## Fabrication

## Packaging and Testing

## Chip

## Partitioning

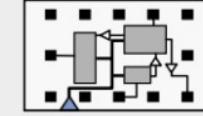
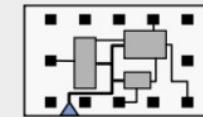
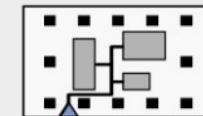
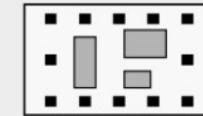
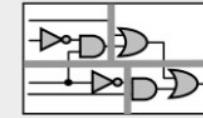
## Chip Planning

## Placement

## Clock Tree Synthesis

## Signal Routing

## Timing Closure



# Various files in Physical Design

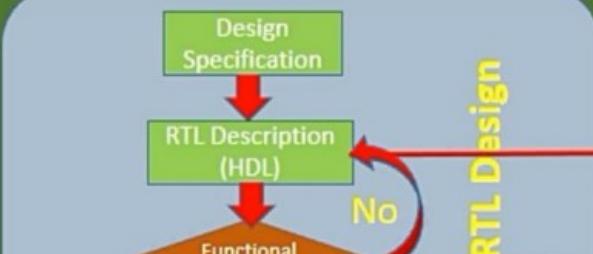
- |                              |                |
|------------------------------|----------------|
| 1) .v file                   | 11) .saif file |
| 2) .vhd file                 | 12) .spf       |
| 3) .lib file                 | 13) .sbpf      |
| 4) .db file                  | 14) .sdf       |
| 5) .lef file                 | 15) .map       |
| 6) .tf file                  | 16) .itf       |
| 7) .mw file                  | 17) .io file   |
| 8) .def file                 | 18) .tcl file  |
| 9) .sdc file                 | 19) .rspf      |
| 10) .tlu file (TLU+ Library) | 20) Simv       |
|                              | 21) .gds file  |

## RTL Design

## Logic Design

## Physical Design

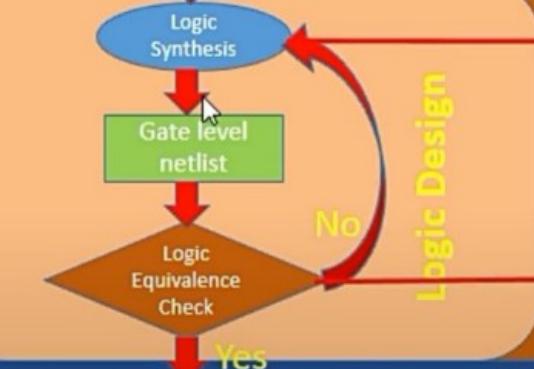
### Front-End Design



```
module mux2to1(Data_in_0, Data_in_1, sel, Data_out);  
    input Data_in_0;  
    input Data_in_1;  
    input sel;  
    output Data_out;  
    reg Data_out;  
    always @([Data_in_0, Data_in_1, sel]) begin  
        if(sel == 0)  
            Data_out = Data_in_0;  
        else  
            Data_out = Data_in_1;  
    endendmodule
```

Synopsys VCS  
Mentor Graphics Questasim  
Xilinx Vivado

### Back-End Design



Design Compiler –Synopsys  
Genus – Cadence  
LeonardoSpectrum – Mentor Graphics

Formality –Synopsys  
Conformal – Cadence  
QuestaSLEC– Mentor Graphics



Innovus - Cadence  
IC Compiler – Synopsys  
Olympus SOC– Mentor Graphics

# Calibre Design Solutions Portfolio

## Advanced Physical Verification



Calibre  
nmDRC

Calibre  
MP

Calibre  
Pattern Match

Calibre  
Auto-Waivers

## Design for Manufacturing

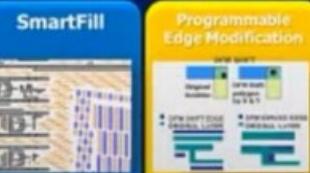
Market

What it Does

46 3D5, IC Overview

### Calibre YieldEnhancer

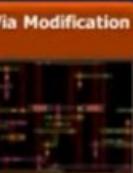
SmartFill



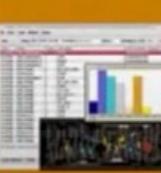
Programmable Edge Modification



Via Modification



### YieldAnalyzer



### Calibre LFD



Market

IC Design Below 20nm  
Analog > 20nm

All Submicron IC  
Design

All Submicron IC  
Design

IC Design Below  
40nm

IC Design Below  
40nm

What it Does

Insertion of  
Fill Geometry

Automated Layout  
Modification

Via  
Yield Improvement

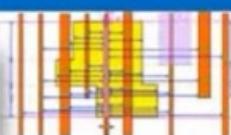
Critical Area Analysis /  
DFM Scoring

Litho Verification/  
ModelBasedHints

## Circuit Verification



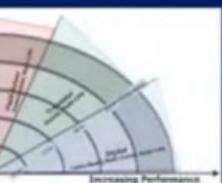
### Calibre nmLVS



### Calibre PERC



### Calibre xACT



All Submicron IC  
Design

Parasitic  
Extraction

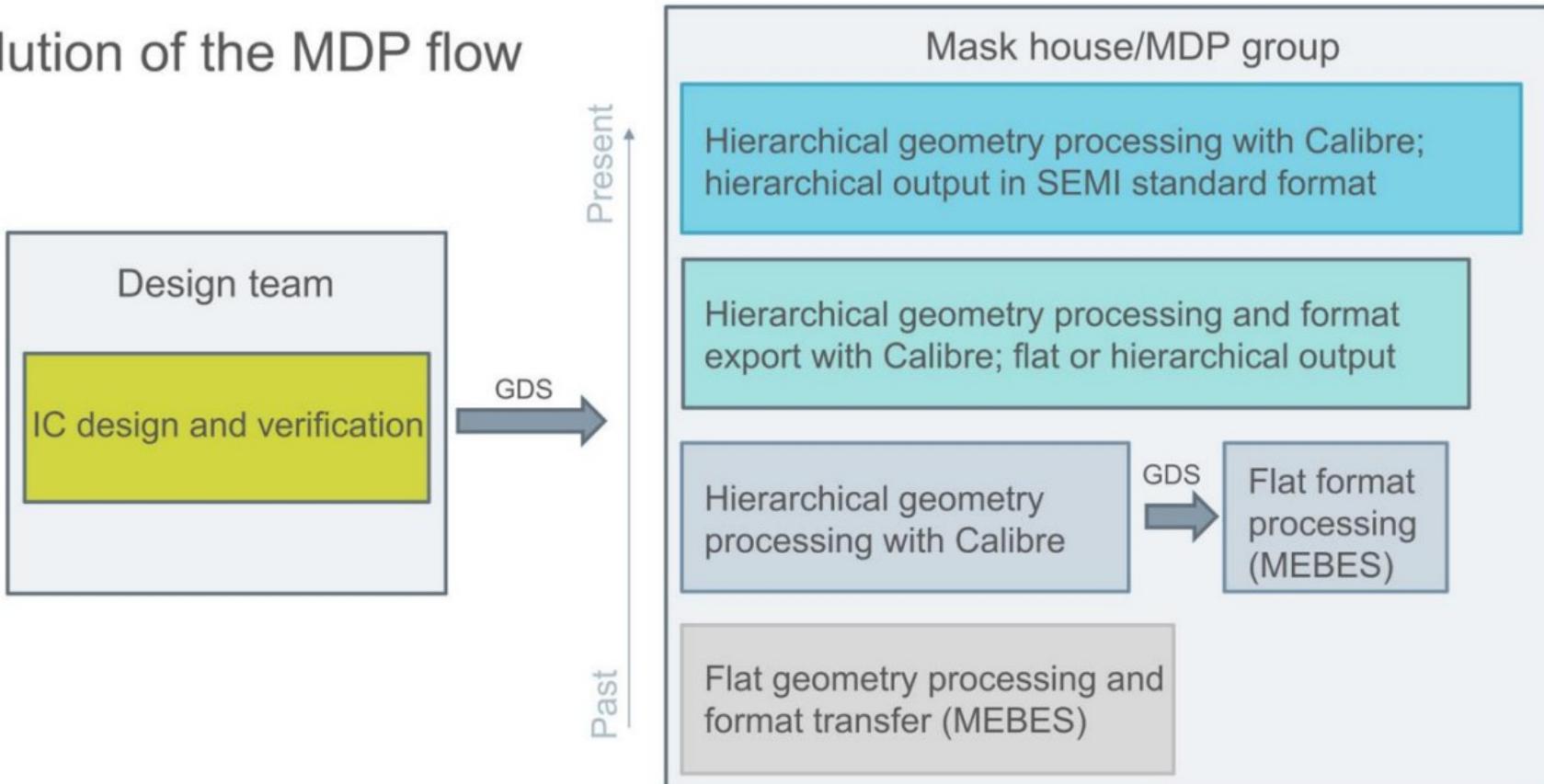
Mentor  
Graphics



# Mask data preparation

(*Calibre MDP*)

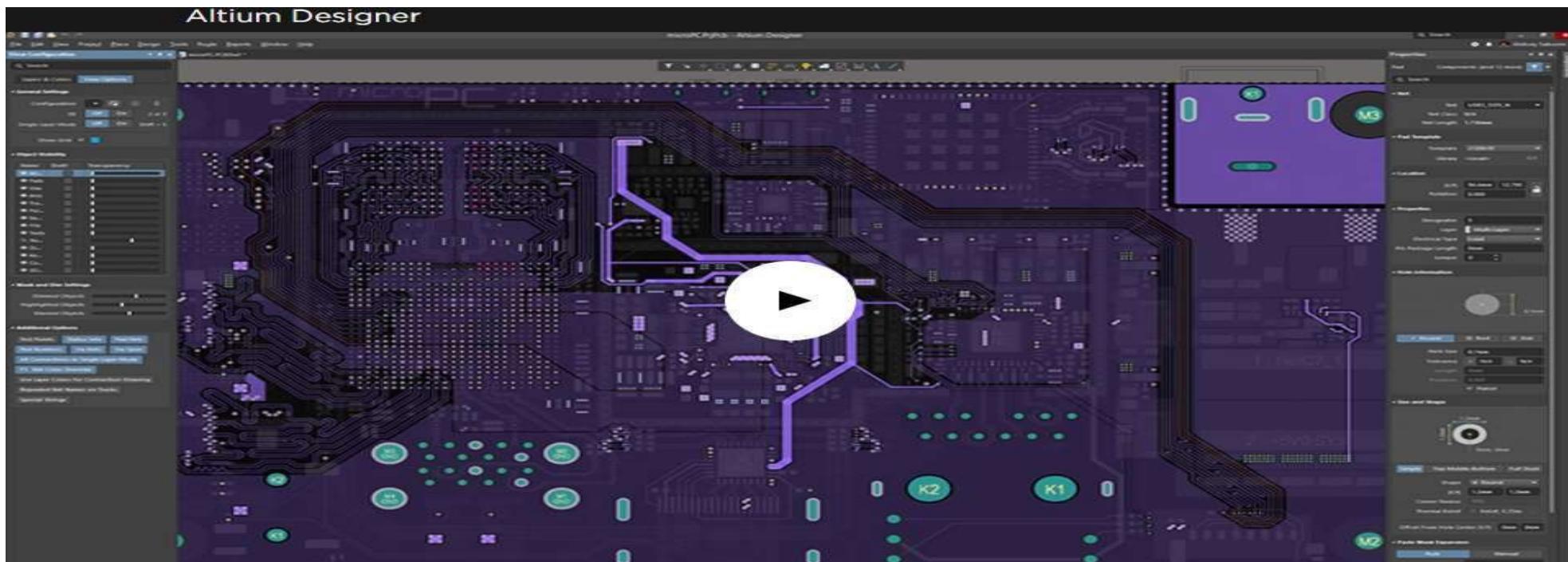
## Evolution of the MDP flow



# DFM

- The design methodology called Design for Manufacturability (DFM) includes a set of techniques to modify the design of ICs in order to make them more manufacturable, i.e. to improve their functional yield, parametric yield, or reliability.

# Проектування комп'ютерних систем на рівні топології друкованої плати



# Рівні проектування цифрових систем

SW: Application		A
SW: C++ Boost		
SW: C++ STL		
SW: C++11		
SW: C++		
SW: C		
SW: Assembler	<b>SW(SSW + SW)</b>	
<hr/>		
(TLM) (ESL) HW: SystemC/SystemVerilog	(Architecture, CE)	<b>HW</b>
(RTL) HW: VHDL/Verilog	(Logic Circuit, CE)(1)	
HW: VHDL/Verilog	(Logic Circuit, CE)(2)	
HW: SPICE/Verilog	(Physical Design, EE)(1)	
HW: SPICE	(Physical Design, EE)(2)	
HW: SPICE/GDSII	(Physical Design, EE-PD) <b>HW (tape-in)</b>	
<hr/>		
(DFM) HW: GDSII	(Physical Design, EE-PD) <b>HW (tape-out)</b>	

**HW (LC)****HW (PD)****МВ****SW****SSW**

1.1 Основи організації та функціонування комп'ютерів

1.2 Комп'ютерна логіка, частина 1

2.1 Комп'ютерна логіка, частина 2

2.2 Комп'ютерна схемотехніка

3.1 Архітектура комп'ютерів

3.2 Комп'ютерні системи

4.1

4.2 Мікропроцесорні системи

Дискретна математика

Теорія електричних та магнітних кіл

Комп'ютерна електроніка

Основи алгоритмізації та програмування (П.ч.1)

Об'єктно-орієнтоване Програмування (П.ч.2)

Структури даних та алгоритми (П.ч.3)

Алгоритми та методи обчислень

**C/C++**

Паралельні та розподілені обчислення

Комп'ютерні мережі

Засоби системного програмування

Системне програмування

Системне програмне забезпечення

Обробка сигналів

**C /Assembler**

<u><a href="#">GitHub</a></u>	<u><a href="#">StackOverflow</a></u>	<u><a href="#">RedMonk</a></u>
<ol style="list-style-type: none"> <li>1. JavaScript.</li> <li>2. Java.</li> <li>3. Python.</li> <li>4. PHP.</li> <li>5. C++.</li> <li>6. C#.</li> <li>7. TypeScript.</li> <li>8. Shell.</li> <li>9. C.</li> <li>10.Ruby.</li> </ol>	<ol style="list-style-type: none"> <li>1. Python.</li> <li>2. JavaScript.</li> <li>3. Java.</li> <li>4. C#.</li> <li>5. PHP.</li> <li>6. C++.</li> <li>7. R.</li> <li>8. SQL.</li> <li>9. Swift.</li> <li>10.C.</li> </ol>	<ol style="list-style-type: none"> <li>1. JavaScript.</li> <li>2. Java.</li> <li>3. Python.</li> <li>4. PHP.</li> <li>5. C#.</li> <li>6. C++.</li> <li>7. CSS.</li> <li>8. Ruby.</li> <li>9. C.</li> <li>10.Objective-C.</li> </ol>
<u><a href="#">StackOverflow</a></u>	<u><a href="#">RedMonk</a></u>	<u><a href="#">dou.ua</a></u>
<ol style="list-style-type: none"> <li>1. Python.</li> <li>2. JavaScript.</li> <li>3. Java.</li> <li>4. C#.</li> <li>5. PHP.</li> <li>6. C++.</li> <li>7. R.v</li> <li>8. SQL.</li> <li>9. Swift.</li> <li>10.C.</li> </ol>	<ol style="list-style-type: none"> <li>1. JavaScript.</li> <li>2. Java.</li> <li>3. Python.</li> <li>4. PHP.</li> <li>5. C#.</li> <li>6. C++.</li> <li>7. CSS.</li> <li>8. Ruby.</li> <li>9. C.</li> <li>10.Objective-C.</li> </ol>	<ol style="list-style-type: none"> <li>1. Java.</li> <li>2. JavaScript.</li> <li>3. C#.</li> <li>4. PHP.</li> <li>5. Python.</li> <li>6. C++.</li> <li>7. TypeScript.</li> <li>8. Swift.</li> <li>9. Ruby.</li> <li>10.Kotlin.</li> <li>11.Go.</li> <li>12.C.</li> <li>13.Scala.</li> <li>14.1C.</li> <li>15.Pascal/Delphi.</li> <li>16.T-SQL.</li> <li>17.PL-SQL.</li> <li>18.Objective-C.</li> <li>19.Groovy.</li> <li>20.Erlang.</li> <li>21.Apex.</li> </ol>

Все категории

Должность, язык, компания, город, страна. Например, Node.js relocation

Найти

 искать в описаниях вакансийБыстрый переход: [Дизайн](#), [начинающим](#), [удаленная работа](#), [работа за рубежом](#).**Java** 350

- [Java Engineer](#) в Symphony Solutions
- [Java Software Engineer](#) в WEBXLOO
- [Junior Java Developer](#) в ControlPay

**.NET** 394

- [Middle Unity C# developer](#)
- Pingle Studio
- [Senior .NET Developer](#) в NIX
- [.NET Software Engineer](#) в Perfectial

**PHP** 369

- [WordPress Developer](#) в SocialTech
- [Junior Backend Developer \(Solid-fintech\)](#) в Genesis
- [Full-Stack developer](#) в PTC Wizard

**C++** 131

- [Team Leader \(C++/Linux\)](#) в Fluent Trade Technologies
- [Junior embedded C Developer](#) в Polaric Semiconductor
- [Senior C++ Software Engineer \[Kyiv\]](#) в EPAM

**Python** 211

- [Full Stack Developer \(Python+JS\)](#) в MWDN
- [Senior Python Engineer #7248](#) в Lohika
- [Python Developer](#) в Perion

**Ruby** 83

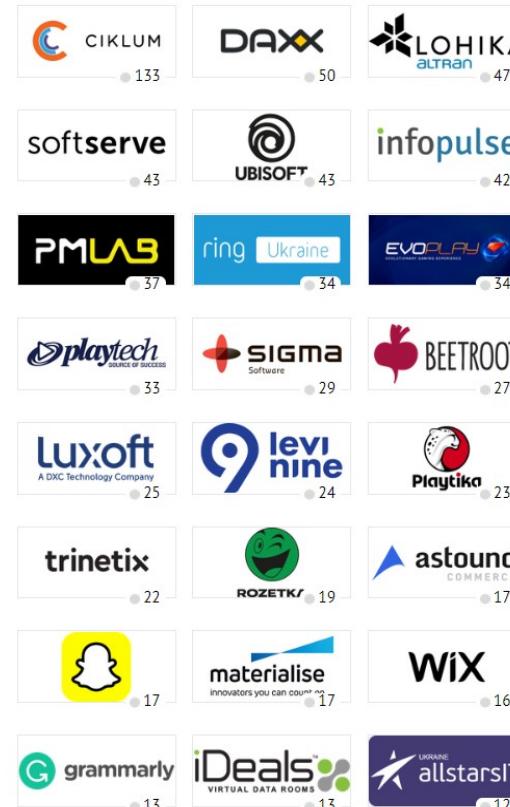
- [Senior Ruby Developer, Parking Booking Platform](#) в DataArt
- [Strong Middle / Senior RoR developer](#) в MasterDynamix
- [Ruby on Rails Developer](#) в AZinac

**iOS/macOS** 90

- [Middle/Senior iOS Developer \(Swift\)](#) ( Toni Laguna Mobile) в Toni Laguna
- [Middle+ iOS Developer](#) в Mind Studios
- [iOS Developer – Calendars 5 team](#) в Readdle

**Android** 123

- [Senior Mobile Developer](#) в Svitla Systems
- [Android developer](#) в TRWA\MORII F

**Вакансии ведущих компаний**

[Вакансии](#)[Тренды](#)[Компании](#)[Рейтинг](#)[Топ-50](#)[Отзывы](#)[Добавить компанию](#)

C++

Должность, язык, компания, город, страна. Например, HR Одесса

 искать в описаниях вакансийБыстрый переход: [Android](#), [начинающим](#), [удаленная работа](#), [работа за рубежом](#).

## 131 вакансия в категории C++ [RSS](#)

 [C++ Developer \(MT4, MT5\)](#) в  Overonix Technologies  Киев

Trading ecosystem which includes such services as: Trading platform, CRM, BI (Reporting System), Roles/Permissions, Chat System, Client Engagement, Dialling system, etc.

 [C++ Developer](#) в  Smart Group-ST \$3000-4000  Киев

We're looking for experienced C++ Developer to join our team. We offer non-trivial tasks and participation in an interesting and complicated long-term project being developed from scratch

 [Senior C++ Engineer with Computer Vision](#) в  Svitla Systems, Inc.  Киев, Харьков,

Львов, удаленно

Svitla Systems Inc. is looking for Senior C++ Engineer for a full-time position (40 hours per week) in Ukraine.

[Senior C++ Software Engineer \[Kyiv\]](#) в  EPAM  Киев

Our customer is a top mass media and information company working on a global market. As part of this project, we make thorough audit and re-architecting of the current outdated solution to archive better code quality, platform-independence and, performance.

[Senior C Software Engineer \(ID 50632\)](#) в  SoftServe  Львов

WE ARE The global leader in labeling and packaging materials and solutions. The company's applications and technologies are an integral part of products used in every major market and industry.

### Опыт

< 1 года

1...3 года

3...5 лет

5+ лет

### Город

Киев 84

Харьков 20

Одесса 15

Львов 10

Днепр 5

Черновцы 1

Запорожье 1

Винница 1

Ровно 1

Ивано-Франковск 1

удаленная работа 12

за рубежом 2

історичні етапи розвитку	рік
Мова BCPL	1966
Мова Бі (оригінальна розробка Томсона для UNIX)	1969
Мова Сі	1972
С із класами	1980
C84	1984
Cfront (випуск Е)	1984
Cfront (випуск 1.0)	1985
Множинне/віртуальне наслідування	1988
Узагальнене програмування (шаблони)	1991
ANSI C++ / ISO-C++	1996
ISO/IEC 14882:1998	1998
ISO/IEC 14882:2003	2003
C++/CLI	2005
TR1	2005
C++11	2011
C++14	2014
C++17	2017
C++20	2020

C.B.

H.E

# Сортування

## //STL(C++)

```
template< class ExecutionPolicy, class RandomIt, class Compare >
void sort( ExecutionPolicy&& policy, RandomIt first, RandomIt last, Compare comp );
```

## //JCL(Java)

```
public static <T> void sort(T[] a, Comparator<? super T> c);
```

## //FCL(C#)

```
public static <T> List<T> sort(List<T> list, String sortByProperty);
```

## Класичне процедурне програмування

### //C, C++

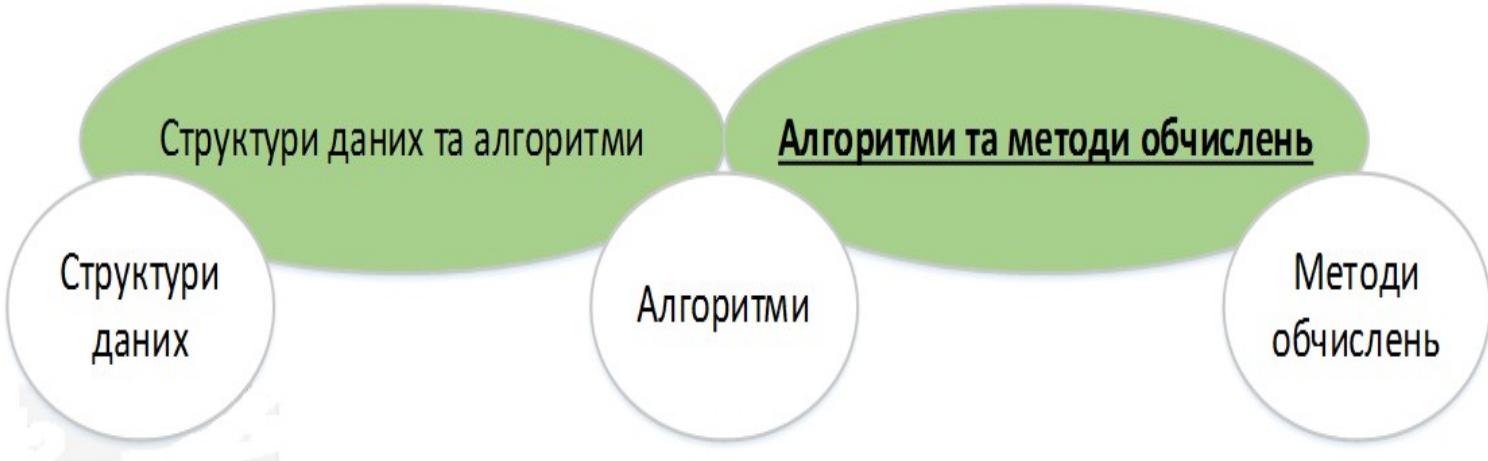
```
void qsort (void* base, size_t num, size_t size, int (*compar)(const void*,const void*));
```

## Пробні контрольні завдання

29. Написати програму додавання масивів з використанням std::vector. Використати ітератори та контейнери.
30. Написати програму пошуку заданої стрічки в іншій стрічці. Стрічки задані як std::string.

**Алгоритми та моделі обчислень**  
*Mісце дисципліни в системі дисциплін спеціальності*  
**«Комп’ютерна інженерія»**

<u>HW (PD)</u>	<u>MB</u>	<u>SW</u>	<u>SSW</u>
1.1		Основи алгоритмізації та програмування (П.ч.1)	
1.2	Дискретна математика	Об'єктно-орієнтоване Програмування (П.ч.2)	
2.1 Д.С.М.С.	Теорія електричних та магнітних кіл Комп'ютерна електроніка	Структури даних та алгоритми (П.ч.3)	
2.2 С.М.С.		Алгоритми та методи обчислень	Засоби системного програмування
3.1 не Ж.В.М.			Системне програмування
3.2 Ж.В.М.		Паралельні та розподілені обчисlenня	Системне програмне забезпечення
4.1			
4.2			
Б.Ж.В.М. 5...			



# Алгоритми та моделі обчислень

Зміст дисципліни

# Алгоритми та моделі обчислень

Основу дисципліни «Алгоритми та методи обчислень» становлять:

Теорія алгоритмів (в зарубіжній літературі фігурує термін теорія обчислень - англ. Theory of Computation). Тут розглядаються:

Теорія автоматів (англ. Automata theory)

Теорія формальних мов (англ. Formal language theory)

Теорія обчислюваності (англ. Computability theory)

Теорія складності обчислень (англ. Computational complexity theory)

Моделі обчислень (англ. Models of Computation).

Методи розробки алгоритмів(алгоритмічні стратегії)(англ. Algorithm Design Paradigm, Algorithmic Paradigm, Algorithmic Technique, Algorithmic Strategy).

Додатково до курсу включений розділ присвячений квантовим обчисленням.

# Алгоритми та моделі обчислень

Для базових алгоритмів обробки інформації та бібліотек популярних мов програмування, які їх реалізовують, розглядаються зразки коду програм, в яких велику увагу приділено використанню:

узагальненого програмування;  
метапрограмування;  
регулярних виразів та нотації Бекуса-Наура.  
Також окрім імперативного(в основному процедурного та об'єктно-орієнтованого) програмування, в наведених зразках коду показано застосування:

парадигми функційного програмування;  
парадигми реактивного програмування;  
парадигми подійно-орієнтованого програмування.

# Алгоритми та моделі обчислень

## Література:

- 1) Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford. Introduction to Algorithms. — 3rd. — MIT Press, 2009. — ISBN 0-262-03384-4.
- 2) Donald E. Knuth. The Art of Computer Programming, Volumes 1-4A Boxed Set. Third Edition (Reading, Massachusetts: Addison-Wesley, 2011), 3168pp. ISBN 978-0-321-75104-1, 0-321-75104-3.
- 3) Michael Sipser (2013). Introduction to the Theory of Computation. 3rd. Cengage Learning. ISBN 978-1-133-18779-0
- 4) Savage, John E. (1998). Models Of Computation: Exploring the Power of Computing. ISBN 978-0-201-89539-1
- 5) Fernandez, Maribel (2009). Models of Computation: An Introduction to Computability Theory. Undergraduate Topics in Computer Science. Springer. ISBN 978-1-84882-433-1.
- 6) Anany Levitin (2012). Introduction to the design & analysis of algorithms. 3rd. ISBN-13: 978-0-13-231681-1
- 7) <https://ocw.mit.edu/courses/mathematics/18-404j-theory-of-computation-fall-2006/>
- 8) <https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-045j-automata-computability-and-complexity-spring-2011/>

# **Частина 1. Тема №1. Поняття моделі обчислень та алгоритму.**

- Історія поняття алгоритму.
- Хронологія теорії алгоритму.
- Визначення алгоритму.
- Основні вимоги до алгоритму.
- Основні етапи повної побудови алгоритму.
- Методи представлення алгоритму.
- Методи проектування алгоритму.
- Оцінка коректності алгоритму.
- Введення в теорію алгоритмів.
- Формалізація поняття алгоритму.

# **Частина 1. Тема №2. Методи синтезу та способи відображення алгоритмів. Підходи при синтезі алгоритмів(алгоритмічні стратегії).**

- Графічне відображення алгоритму.
- Відображення алгоритму за допомогою ПГА.
- Метод “розділяй і пануй”.
- Повний перебір.
- Динамічне програмування.
- Скупі алгоритми.
- Бектрекінг (перебір з поверненням).
- Метод гілок і границь.
- Ймовірністні алгоритми.
- Алгоритми локального пошуку.

# **Частина 1. Тема №3. Основи аналізу алгоритмів.**

- Оцінка розміру вхідних даних.
- Складність по пам'яті.
- Складність по часу виконання алгоритму.
- Порівняння найкращих, середніх та найгірших оцінок.
- Ріст функцій.
- $O$ -,  $\Theta$ -,  $W$ -,  $\omega$ -,  $Q$ - нотації.
- Стандартні класи ефективності алгоритмів.
- Математичний аналіз нерекурсивних алгоритмів.
- Рекурсивні алгоритми.
- Математичний аналіз рекурсивних алгоритмів.
- Емпіричний аналіз алгоритмів.

# **Частина 1. Тема №4. Базові алгоритми обробки інформації.**

## **4.1. Алгоритми пошуку.**

- Послідовний пошук
- Бінарний пошук.
- Пошук в лінійних списках.
- Задача вибору.
- Дерева бінарного пошуку.
- Збалансовані дерева.
- Вичерпний пошук.
- Хешування.
- Розв'язання колізій при хешуванні відкритою адресацією та методом ланцюжків.
- Порозрядний пошук.
- Зовнішній пошук.

# **Частина 1. Тема №4. Базові алгоритми обробки інформації.**

## **4.2. Алгоритми сортування даних.**

- Сортування вибором.
- Сортування вставками.
- Сортування обміном.
- Сортування злиттям.
- Сортування Шелла.
- Швидке сортування.
- Піраміdalne сортування.
- Порозрядне та бітове сортування.
- Методи сортування спеціального призначення.
- Мережі сортування.
- Зовнішнє сортування.

# **Частина 1. Тема №4. Базові алгоритми обробки інформації.**

## **4.3. Алгоритми порівняння зі взірцем.**

- Алгоритм Рабіна-Карпа.
- Пошук підрядків за допомогою скінчених автоматів.
- Алгоритм Кнута-Морріса-Пратта.
- Алгоритм Бойєра-Мура.
- Наближене порівняння рядків.

# **Частина 1. Тема №4. Базові алгоритми обробки інформації.**

## **4.4. Чисельні алгоритми.**

- Матриці та дії з ними.
- Множення матриць по Винограду та по Штрассену.
- Робота з довгими числами.
- Алгебраїчні системи.
- Розв'язок систем лінійних рівнянь.
- Розв'язання нелінійних рівнянь.
- Многочлени та швидке перетворення Фур'є.
- Алгоритми апроксимації і інтерполяція чисельних функцій.

# Частина 1. Тема №4. Базові алгоритми обробки інформації.

## 4.5. Графи та мережеві алгоритми.

- Пошук у графі.
- Породження всіх каркасів графа.
- Каркас мінімальної ваги.
- Метод Дж. Краскала.
- Метод Р. Пріма.
- Досяжність. Визначення зв'язності. Двозв'язність.
- Ейлерові цикли.
- Гамільтонові цикли.
- Фундаментальна множина циклів.
- Алгоритм Дейкстри
- Алгоритм Флойда.
- Метод генерації всіх максимальних незалежних множин графа.
- Задача про найменше покриття.
- Задача про найменше розбиття.
- Розфарбування графа.
- Пошук мінімального розфарбування вершин графа.
- Використання задачі про найменше покриття при розфарбуванні вершин графа.
- Потоки в мережах.
- Метод побудови максимального потоку в мережі.
- Методи наближеного рішення задачі комівояжера(метод локальної оптимізації, алгоритм Эйлера, алгоритм Кристофідеса).
- Аналіз алгоритм на графах.

# **Частина 1. Тема №4. Базові алгоритми обробки інформації.**

## **4.6. Паралельні та розподілені алгоритми.**

- Модель паралельного виконання програми зі спільною пам'яттю і модель передачі повідомень.
- Організація паралельних обчислень відповідно до принципу консенсусу і на основі вибору.
- Методи визначення завершення паралельних обчислень.
- Паралельний пошук.
- Паралельне сортування.
- Паралельні чисельні алгоритми.
- Паралельні алгоритми на графах.

# **Частина 1. Тема №5. Бібліотеки основних алгоритмів обробки інформації для популярних мов програмування. (6 годин)**

- 5.1. Застосування базових алгоритмів при узагальненому програмуванні на C++ засобами STL(Standart Template Library).
- 5.2. Застосування базових алгоритмів при узагальненому програмуванні на Java засобами JCL(Java Class Library).
- 5.3. Застосування базових алгоритмів при узагальненому програмуванні на C# засобами FCL(Framework Class Library).
- 5.4. Застосування алгоритмів лінійної алгебри при програмуванні на C++ засобами uBLAS.
- 5.5. Застосування алгоритмів обробки сигналів при програмуванні на C++ засобами OpenCV(Open Source Computer Vision Library).

# Частина 2. Тема 6. Моделі обчислень. (8 годин)

6.1. Послідовні формальні алгоритмічні системи еквівалентні машині Тюрінга. Теза Черча.

- 6.1.1. Нормальні алгоритми Маркова.
- 6.1.2. Регістрова машина.
- 6.1.3. РАМ-машина.

# Частина 2. Тема 6. Моделі обчислень. (8 годин)

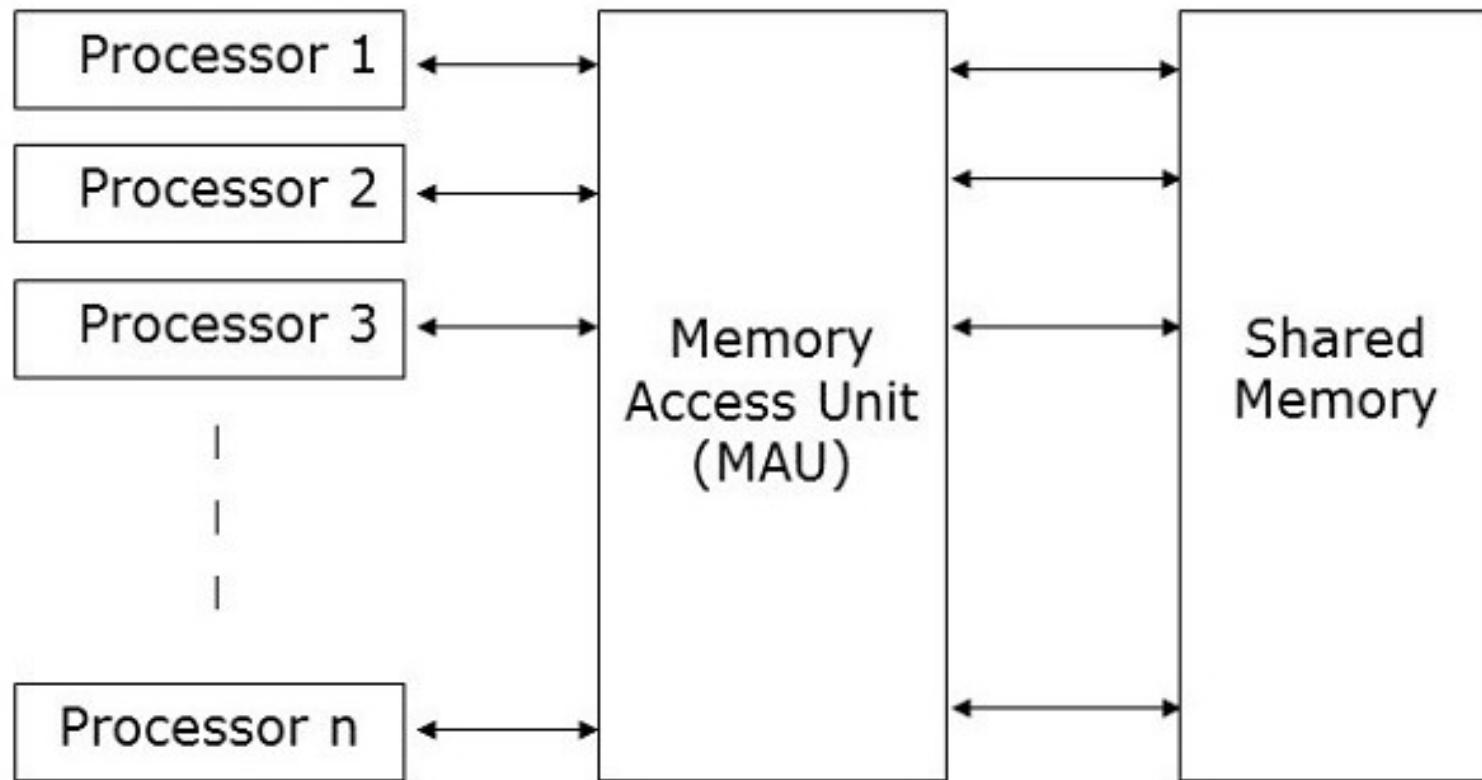
6.2. Функційні моделі обчислень та парадигма функційного програмування.

- 6.2.1. Функційні моделі обчислень.
- 6.2.1.1. Лямбда числення.
- 6.2.1.2. Типоване лямбда числення.
- 6.2.1.3. Рекурсивні функції.
- 6.2.1.4. Комбінаціна логіка(як функційна модель обчислень).
- 6.2.1.5. Клітковий автомат.
- 6.2.1.6. Абстрактна перезаписуюча система.
- 6.2.2. Парадигма функційного програмування

# Частина 2. Тема 6. Моделі обчислень. (8 годин)

6.3. Паралельні моделі обчислень та парадигма реактивного програмування. Паралельне програмування.

- 6.3.1. Паралельні моделі обчислень.
- 6.3.1.1. ПРАМ-машина. ПРАМ-машина на основі ПВДН.
- 6.3.1.2. Мережа процесів Кана.
- 6.3.1.3. Мережа Петрі.
- 6.3.1.4. Мережа взаємодій.
- 6.3.1.5. Синхронний потік даних.
- 6.3.2. Парадигма реактивного програмування.
- 6.3.3. Паралельне програмування



# Частина 3. Тема №7. Квантові обчислення. (2 години)

7.1. Основні поняття і принципи квантових обчислень. Квантова машина Тюрінга і BQP-клас складності.

7.2. Кубіти, квантові вентилі та квантові регістри.

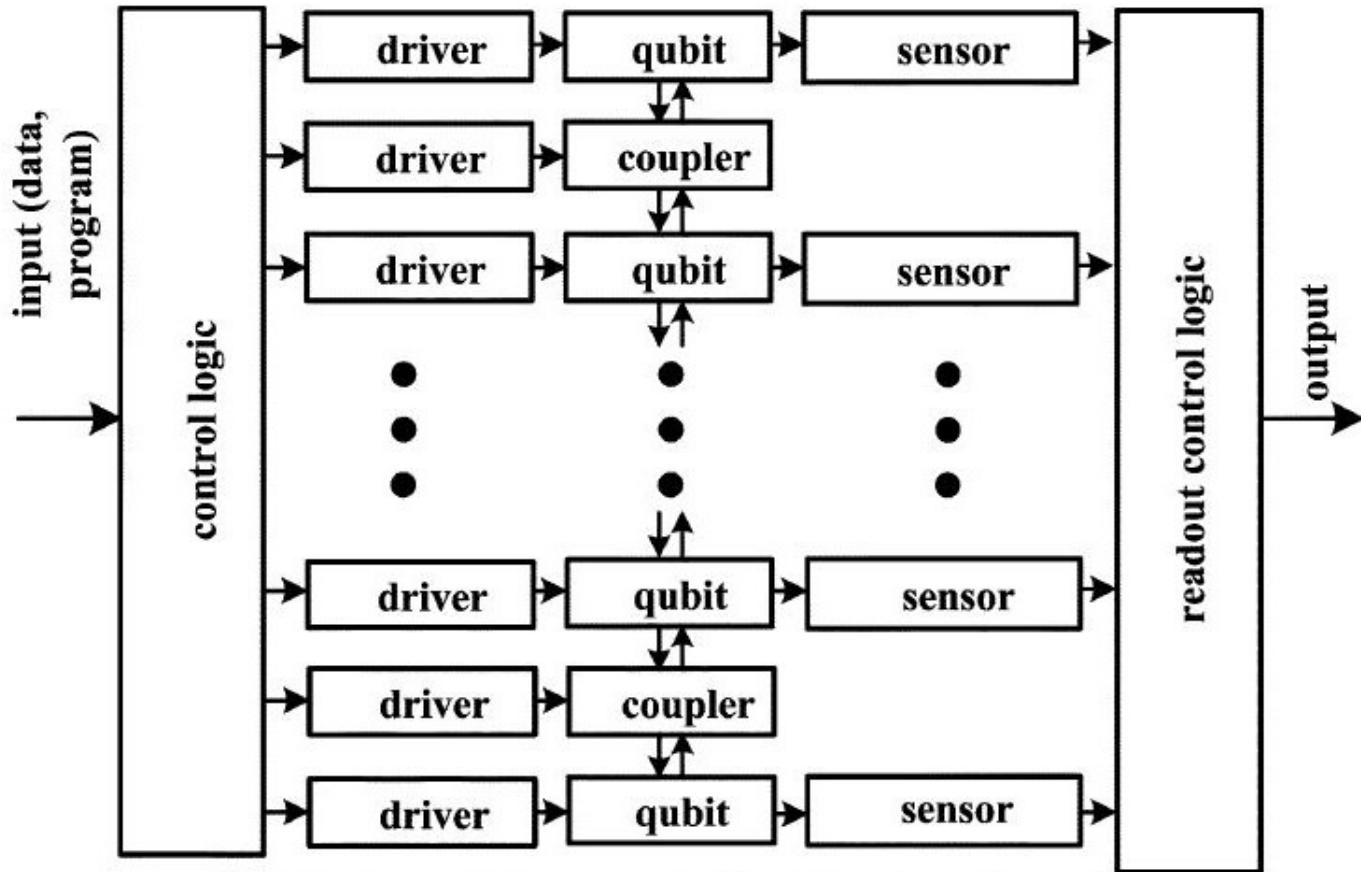
Вимірювання значень кубітів. Система кубітів та квантова запутаність. Вентиль тотожного перетворення. Вентиль заперечення. Вентиль фазового зміщення. Вентиль перетворення Адамара. Прямий керуючий вентиль. Вентиль контролюваного заперечення. Вентиль Тоффолі. Вентиль Фредкіна.

7.3. Функціонування квантової системи.

Задача Дойча. Задача Дойча-Джозі. Задача Бернштайна-Вазірані. Задача Саймона.

7.4. Важливі алгоритми квантових обчислень.

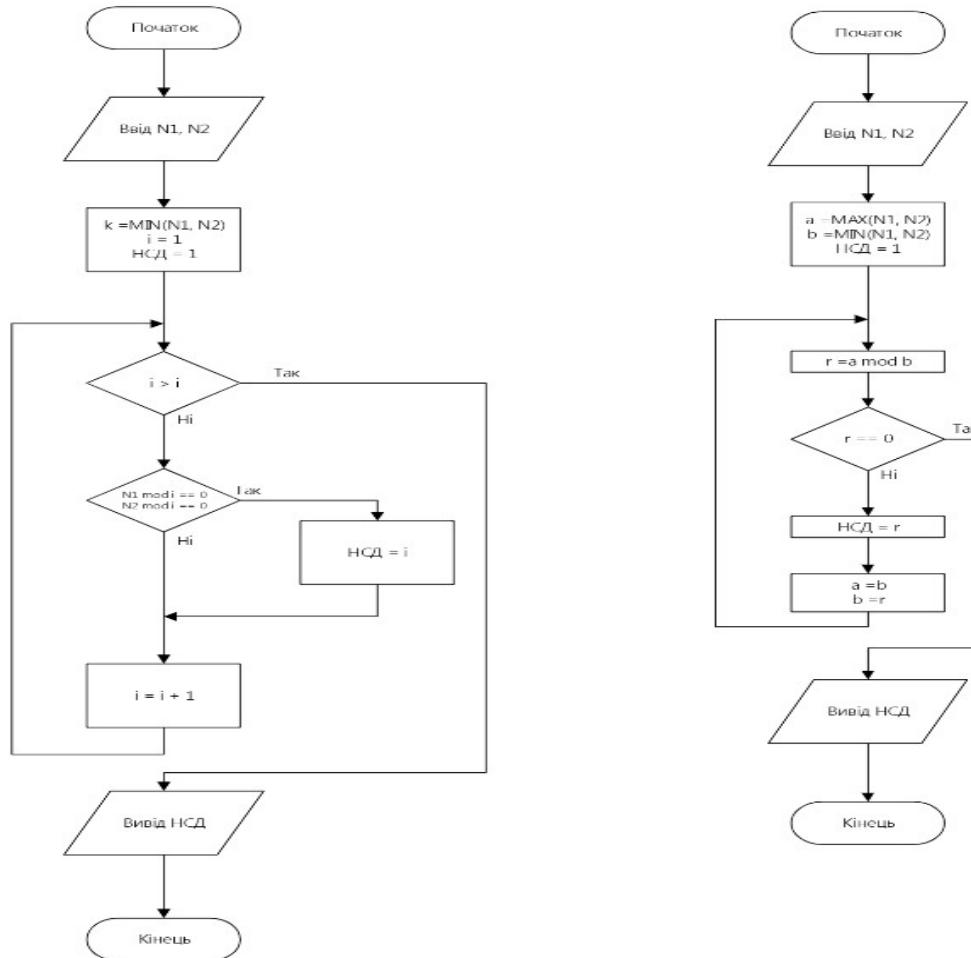
Алгоритм Шора. Алгоритм Гровера.



Architecture of Superconductive Quantum Computer

**Назва роботи:** алгоритм; властивості, параметри та характеристики складності алгоритму.

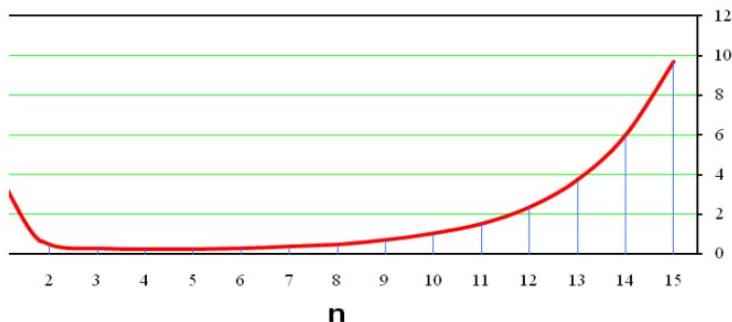
**Мета роботи:** проаналізувати складність заданих алгоритмів.



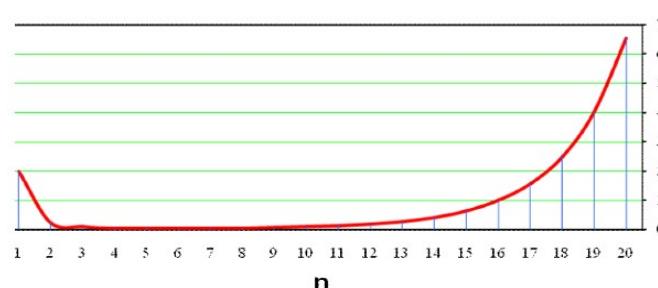
**Назва роботи:** асимптотичні характеристики складності алгоритму; алгоритми з поліноміальною та експоненціальною складністю.

**Мета роботи:** ознайомитись з асимптотичними характеристиками складності та класами складності алгоритмів.

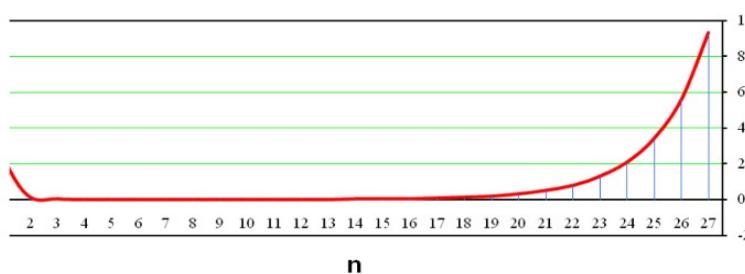
$$Y(n) = \frac{2^n}{n^3} :$$

**K=3**

$$Y(n) = \frac{2^n}{n^4} :$$

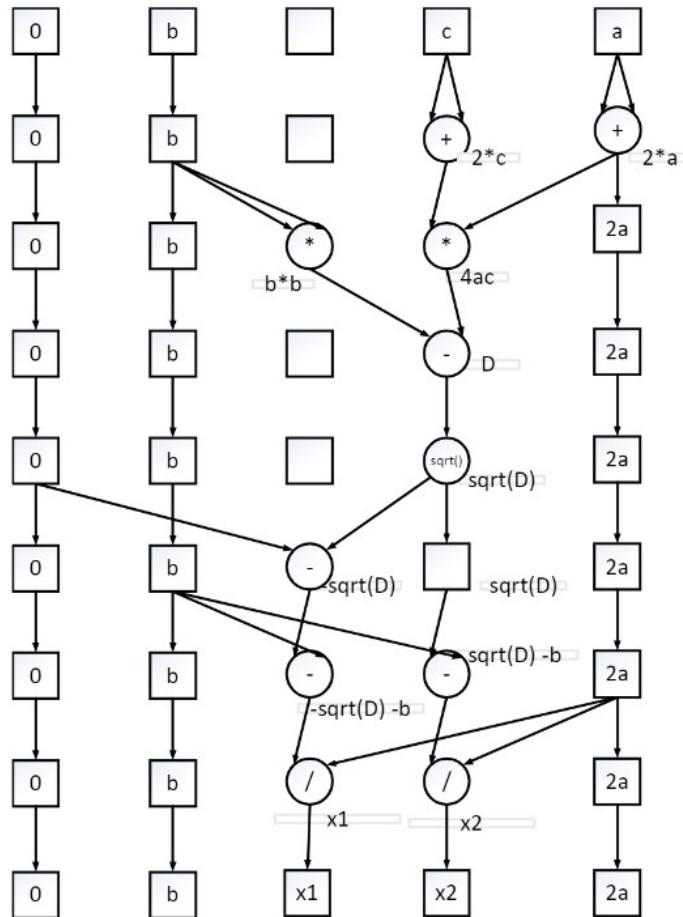
**K=4**

$$Y(n) = \frac{2^n}{n^5} :$$

**K=5**

**Назва роботи:** використання потокового графу алгоритму при проектуванні паралельних обчислень.

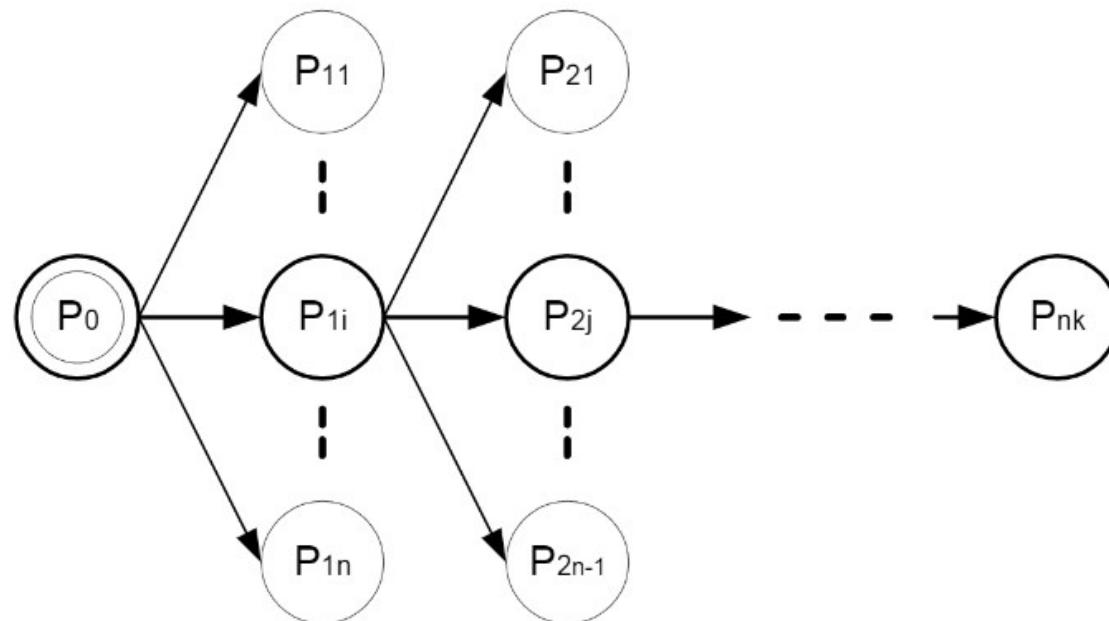
**Мета роботи:** ознайомитися з використанням потокового графу алгоритму при паралельному програмуванні.



## ЛАБОРАТОРНА РОБОТА №4

**Назва роботи:** побудова алгоритмів ефективних за часовою складністю;  
задача квадратичного призначення.

**Мета роботи:** виконати зменшення часової складності методом «гілок і границь».



Задане дискретне робоче поле (ДРП) і матриця зв'язності R. Розташувати елементи X1, X2, X3, X4, X5 на ДРП за критерієм мінімальної сумарної довжини з'єднань.

ДРП


R

x	1	2	3	4	5
1	0	1	2	0	3
2	1	0	3	0	2
3	2	3	0	1	2
4	0	0	1	0	1
5	3	2	2	1	0

Позначимо на ДРП номера позицій в правому верхньому куті вільних позицій та визначемо матрицю відстаней D.

ДРП

1				
	2			
		4	5	
	3			

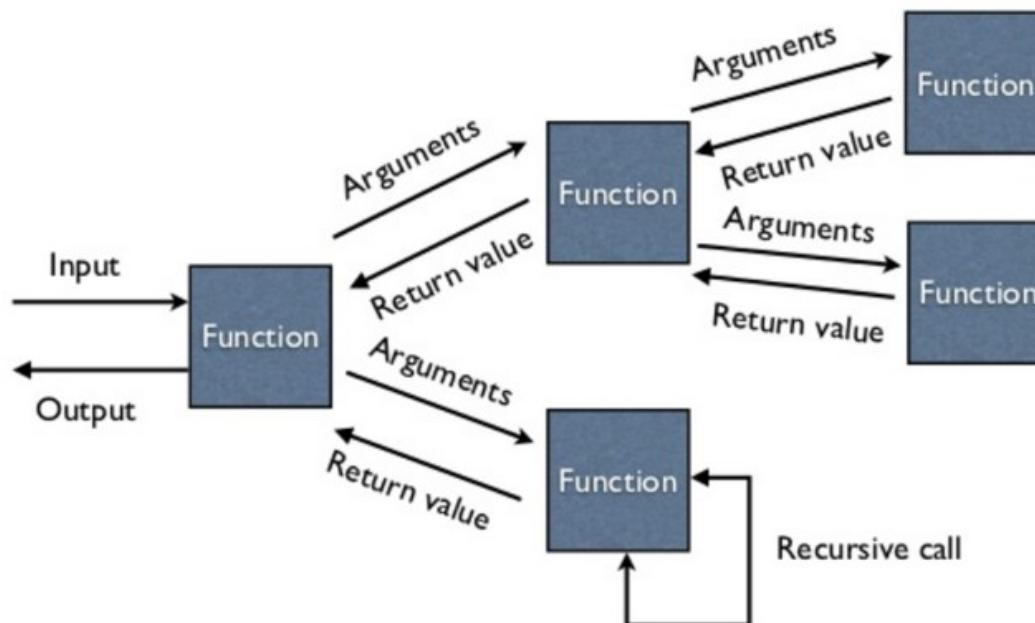
D

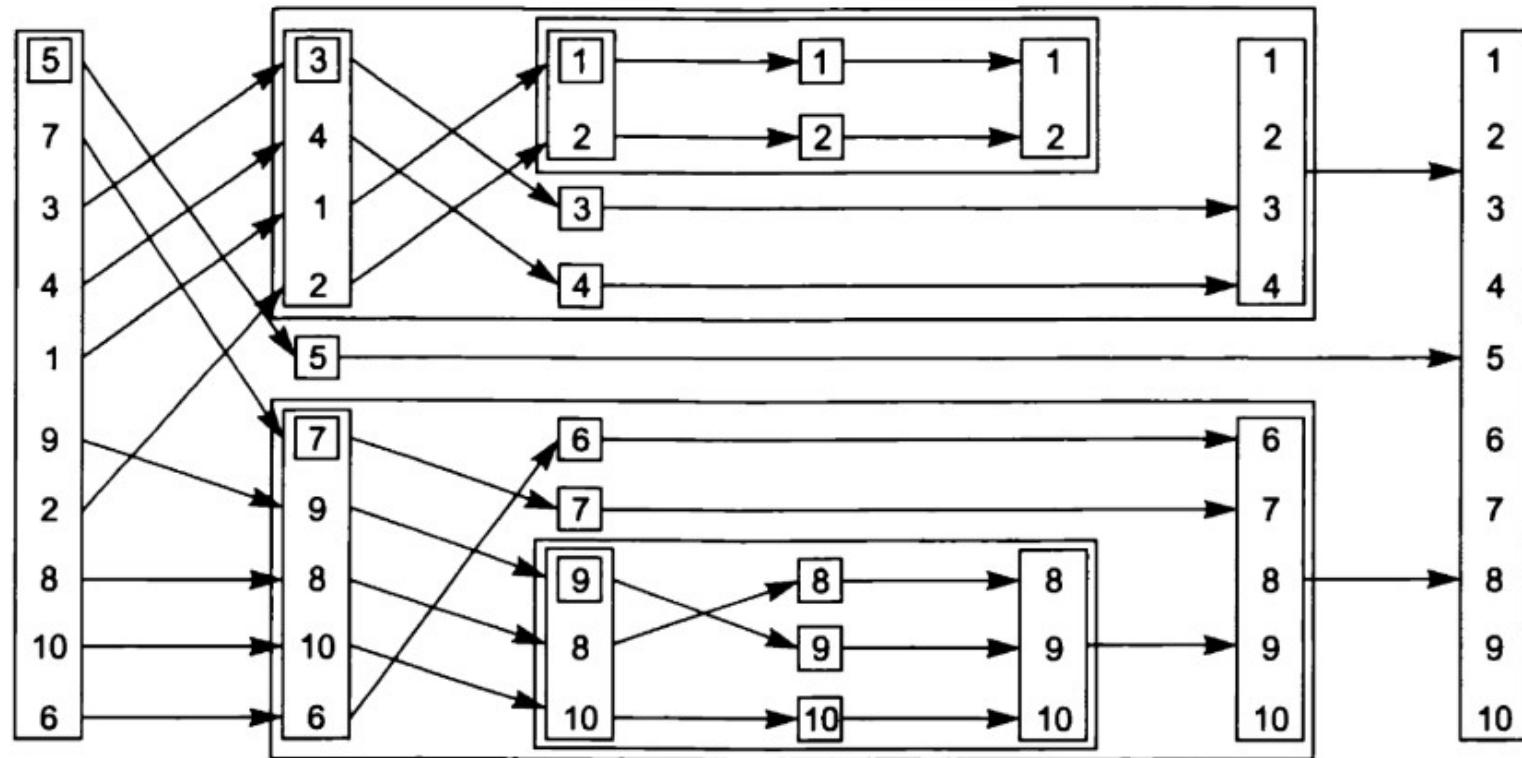
p	1	2	3	4	5
1	0	2	4	4	5
2	2	0	2	2	3
3	4	2	0	2	3
4	4	2	2	0	1
5	5	3	3	1	0

## ЛАБОРАТОРНА РОБОТА №5

**Назва роботи:** функційне програмування.

**Мета роботи:** ознайомитися з парадигмою функційного програмування.

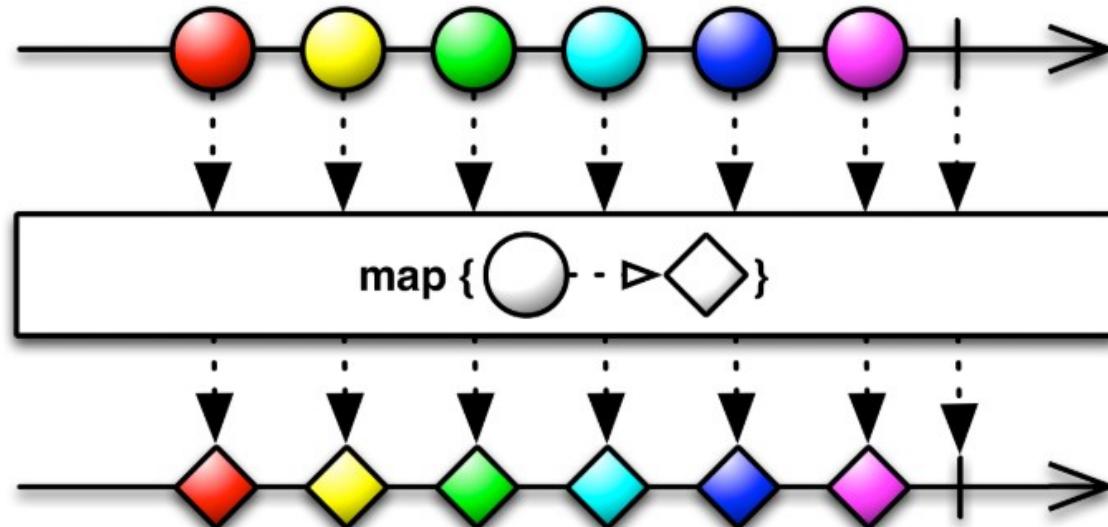




## ЛАБОРАТОРНА РОБОТА №6

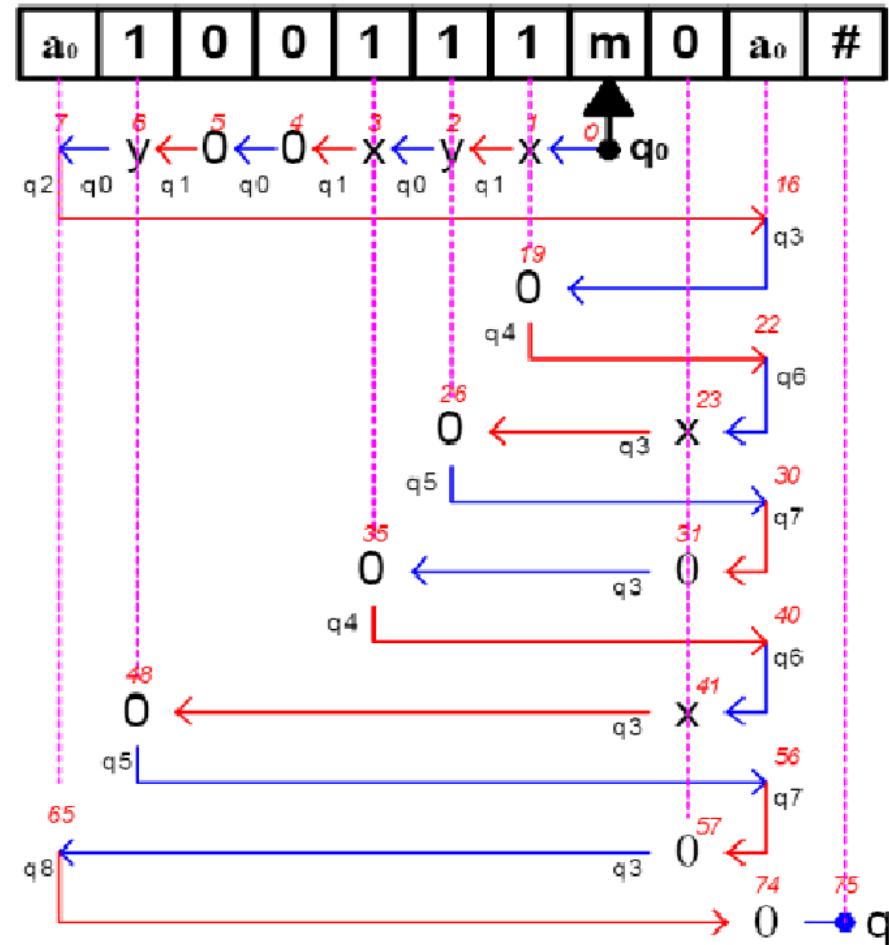
**Назва роботи:** реактивне програмування.

**Мета роботи:** ознайомитися з парадигмою реактивного програмування.



## Практичне заняття № 6

Формальні алгоритмічні системи (ФАС). Машина Тюрінга (МТ).



# Лекційні домашні завдання

# Домашні завдання №1 та №2

(1.2.6. Формальні граматика та формальні мови. (1.3.))

З використанням `std::regex` розробити програму, яка шукає в тексті лексеми заданого формату.

Розробити програму, яка за допомогою `boost::spirit` реалізовує заданий відповідно до варіанту синтаксис для запису виразу.

# Домашні завдання №3 та №4

(1.2.2. Формалізація поняття алгоритму. Формальні алгоритмічні системи(ФАС).)

З використанням реалізації поведінки детермінованого кінцевого автомата написати на C/C++ програму(*англ. automata-based programming*) виявлення ділянки тексту у вхідній стрічці. Програма має лише відображати перехід моделі автомата у кінцевий стан без виявлення позиції входження шуканої ділянки тексту.

Виконати реалізацію моделі машини Тюрінга на C/C++ для виконання обчислень згідно власного варіанту.

# Домашні завдання №5, №7 та №8

## (2. 2. Синтез алгоритмів.)

Скласти програму (C/C++), яка дозволяє знайти невідоме значення  $x$  методом повного перебору для заданих констант  $a, b, c, d, e, f, g$  та  $h$  та порівняти його з обчисленим.

Скласти програму (C/C++), яка з застосуванням алгоритмічної стратегії «перебір з поверненням»(англ. *backtracking*) дозволяє виконати таке розміщення фігур ферзів на шахівниці, що жодна з них не ставить під удар іншу.

Скласти програму (C/C++), яка з застосуванням алгоритмічної стратегії «динамічне програмування» дозволяє виконати такі два завдання:

1. Менеджмент ІТ-компанії розглядає можливість старту нових напрямків. Усі нові напрямки за прибутковістю можна розділити на  $T$  типів. Кожна група певного типу має складатися з  $N_i$  розробників та орієнтовно може приносити компанії прибуток  $K_i$ . Сформувати оптимальну кількість команд відповідно до напрямків, якщо компанії вдалося найняти  $M$  нових розробників.

2. Існує  $T$  номіналів монет. Видати здачу  $M$  мінімальною кількістю монет.

Для порівняння повторно виконати завдання за допомогою «прямого перебору».

# Домашні завдання №12, №9, №6 та №10

(4.1. Алгоритми пошуку. 4.2. Алгоритми сортування даних. + Тема №3)

Скласти програму (C/C++), яка виконує імплементацію хеш-таблиці(*геш-таблиці, англ. Hash table*) без застосування готових реалізацій.

Для алгоритму, що заданий за допомогою функції мовою С, розрахувати обчислювальну складність для «найгіршого випадку», сформувати блок-схему алгоритму та перевірити коректність його роботи.

Скласти програму (C/C++), яка дозволяє відсортувати вхідні величини методом злиття. Безпосередньо сортування методом злиття виконується при об'єднанні частин вхідних даних, які в свою чергу сортуються за допомогою швидкого сортування з стандартної бібліотеки мови С. Кількість частин для злиття та розмір вхідного масиву даних обрати відповідно до варіанту.

Скласти програму (C/C++), яка за допомогою швидкого сортування(з стандартної бібліотеки мови С) дозволяє з вхідного тексту вивести слова, що починаються з заданої літери(решта слів вивести в алфавітному порядку). Сортування потрібно виконати без використання додаткової пам'яті, дозволяється використовувати тільки масив, що зберігає вказівники на початок слів.

# Домашнє завдання №11

## (4.3. Алгоритми порівняння зі взірцем.)

Написати на C/C++ програму, яка реалізовує алгоритм Бояра-Мура для знаходження у вхідній стрічці всіх входжень заданого слова.

# Домашні завдання №13 та №14

## (4.4. Чисельні алгоритми.)

Скласти програму (C/C++), що виконує множення матриць. Для виконання завдання замість простого двовимірного масиву використати власну реалізацію.

Написати на C/C++ реалізацію довгих чисел на основі списку(елементи списку – десяткові цифри) з використанням вказівників. Реалізувати дію згідно варіанту.

# Домашні завдання №15 та №16

## (4.5. Графи та мережеві алгоритми.)

Скласти програму (C/C++), яка дозволяє знаходити мінімальне кістякове дерево(*англ. minimum spanning tree*) для заданого графу за допомогою алгоритму Крускала.

Скласти програму (C/C++), яка дозволяє знаходити у графі мінімальний шлях від заданої вершини до інших вершин за допомогою алгоритму Дейкстри.

Домашні завдання  
№17, №18, №19, №20 та №206

# Домашні завдання №17 та №18

Застосовуючи STL скласти програму (C/C++), яка за допомогою *std::sort* дозволяє з вхідного тексту(*std::string*) вивести слова, що починаються з заданої літери(решта слів вивести в алфавітному порядку). Сортування потрібно виконати без використання додаткової пам'яті, дозволяється використовувати тільки *std::vector*, що зберігає індекси на початок слів. Додатково потрібно зберегти результати роботи програми у *std::map*.

Індекси із *std::vector* вивести на екран за допомогою *std::copy*, а слова за допомогою звичайного циклу *for*. Для виводу даних з *std::map* застосувати *std::transform*.

Замінивши використання *std::vector* на *boost::container::vector*, а *std::map* на *boost::container::map*, виконати домашнє завдання №17 повторно.

# Домашнє завдання №19

Застосовуючи JCL скласти програму (Java), яка за допомогою ***Collections.sort*** дозволяє з вхідного тексту(***String***) вивести слова, що починаються з заданої літери(решта слів вивести в алфавітному порядку). Сортування потрібно виконати без використання додаткової пам'яті, дозволяється використовувати тільки ***ArrayList***, що зберігає індекси на початок слів. Додатково потрібно зберегти результати роботи програми у ***HashMap***.

Застосувати різні способи для виводу результатів роботи програми.

# Домашні завдання №20а та №20б

Застосовуючи FCL скласти програму (C#), яка за допомогою *List<T>.Sort* дозволяє з вхідного тексту(*String*) вивести слова, що починаються з заданої літери(решта слів вивести в алфавітному порядку). Сортування потрібно виконати без використання додаткової пам'яті, дозволяється використовувати тільки *List*, що зберігає індекси на початок слів. Додатково потрібно зберегти результати роботи програми у *Dictionary*.

Застосувати різні способи для виводу результатів роботи програми.

C++/CLI дозволяє писати програми, які одночасно можуть містити і звичайний некерований код(англ. *unmanaged code*) написаний на C++ для компіляції безпосередньо в машинний код, і керований код (англ. *managed code*) написаний на C++ для .NET.

В якості самостійної роботи пропонується виконати домашнє завдання №20 повторно за допомогою C++/CLI.

Домашні завдання №21, №22а та №22б

# Домашнє завдання №21

Написати на C++ програму для розв'язання системи лінійних алгебраїчних рівнянь(СЛАР) з використанням бібліотеки *uBLAS*(з набору бібліотек *Boost*).

СЛАР	$\Rightarrow$	Макровизначення
$\begin{cases} x_1 + 5x_2 + 3x_3 - 4x_4 = 20 \\ 3x_1 + x_2 - 2x_3 = 9 \\ 5x_1 - 7x_2 + 10x_4 = -9 \\ 3x_2 - 5x_3 = 1 \end{cases}$	$\begin{pmatrix} 1 & 5 & 3 & -4 \\ 3 & 1 & -2 & 0 \\ 5 & -7 & 0 & 10 \\ 0 & 3 & -5 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 20 \\ 9 \\ -9 \\ 1 \end{pmatrix}$	
	$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$	#define EQUATIONS_COUNT 4
	$\begin{pmatrix} 1 & 5 & 3 & -4 \\ 3 & 1 & -2 & 0 \\ 5 & -7 & 0 & 10 \\ 0 & 3 & -5 & 0 \end{pmatrix}$	#define COEFFICIENTS \ 1.0, 5.0, 3.0, -4.0,\ \ 3.0, 1.0, -2.0, 0.0,\ \ 5.0, -7.0, 0.0, 10.0,\ \ 0.0, 3.0, -5.0, 0.0
	$\begin{pmatrix} 20 \\ 9 \\ -9 \\ 1 \end{pmatrix}$	#define CONSTANT_TERMS \ 20.0,\ \ 9.0,\ \ -9.0,\ \ 1.0

# Домашні завдання №22а та №22б

Застосовуючи OpenCV скласти програму (Python), яка дозволяє виявляти на фото обличчя людей. При цьому потрібно виявляти не більше MAX\_FACES\_COUNT облич.

В якості самостійної роботи пропонується виконати домашнє завдання №22 повторно за допомогою C++.

Домашні завдання  
№23а, №23б1, №23б2, №23в, №24  
та лабораторна робота №5

# Домашні завдання №23а, №23б1 та №23б2

23 а) Застосовуючи парадигму функційного програмування скласти програму мовою Haskell, яка виконує імплементацію швидкого сортування без використання готових бібліотечних реалізацій.

23 б) Одною з альтернатив Haskell при використанні парадигми функційного програмування є Erlang/Elixir.

Загалом, якщо Haskell можна вважати базовою стандартизованою мовою при застосуванні парадигми функційного програмування, то Erlang та Elixir володіють значною практичною цінністю. Це пов'язано з тим, що для них доступні потужні Web-фреймворки, які необхідні для швидкого написання сучасного масового програмного забезпечення. Також отриманий байт-код після трансляції коду мовою Elixir виконується на віртуальній машині Erlang (BEAM), тому Elixir має сумісність з фреймворком Erlang/OTP та іншими бібліотеками і фреймворками мови Erlang.

В якості самостійної роботи пропонується виконати домашнє завдання №23 повторно за допомогою Erlang або Elixir без використання готових бібліотечних реалізацій.

# Домашні завдання №23в та №24

24 ) Виконати домашнє завдання №23 повторно за допомогою мови Java без використання готових бібліотечних реалізацій.

23 в) Мова Kotlin це намагання спростити програмування для платформи Java.

Байт-код після трансляції коду мовою Kotlin або мовою Scala виконується на віртуальній машині Java (JVM), тому Java/Scala/Kotlin можна віднести до спільної категорії засобів програмування.

Особливою популярністю мова Kotlin користується при створенні Android-додатків, де замість JVM використовується Dalvik/ART.

В якості самостійної роботи пропонується виконати домашнє завдання №23 повторно за допомогою Kotlin без використання готових бібліотечних реалізацій.

Домашні завдання  
№25а, №25б, №25в, №26, №27а, №27б, №28  
та лабораторна робота №6

# Домашнє завдання №25а

Застосовуючи елементи реактивної парадигми програмування(код має містити в явній формі Observable та Observer) за допомогою RxJS(реалізація ReactiveX для JavaScript) скласти програму (мовою JavaScript для платформи NodeJS), яка дозволяє виконувати валідацію ключа(25 шістнадцяткових цифр) ліцензії для деякого програмного продукту. Для зручності вводу програма має групувати шістнадцяткові цифри по п'ять цифр та відображає ‘A’, ’B’, ’C’, ’D’, ’E’ та ’F’ завжди у верхньому регістрі. Okрім шістнадцяткових цифр та символів пробілу і табуляції(що трактуються як одна цифра 0), програма валідації не дозволяє вводити жодних інших символів, які не належать шістнадцятковій системі числення. При цьому надається не більше ATTEMPTS\_COUNT спроб вводу ключа ліцензії. Між ітераціями сусідніх спроб введені значення мають зберігатися для редагування у наступній спробі.

Ключ ліцензії «11111 22222 33333 44444 55555» має міститися в коді програми валідації у неявній формі.

*\* акцентується увага на тому, що завдання має бути виконано за допомогою RxJS відповідно до парадигми реактивного програмування, а не подійно-орієнтованої парадигми, на основі якої працює рушій платформи NodeJS*

# Домашнє завдання №256

Виконати домашнє завдання №25 повторно за допомогою мови C++ використовуючи RxCpp(реалізація ReactiveX для C++).

# Домашнє завдання №25в

Хоча React і не застосовує парадигму реактивного програмування, але в якості ще однієї самостійної роботи пропонується виконати завдання №25 повторно за допомогою React та з використанням Redux.

# Домашнє завдання №26

Виконати домашнє завдання №25 повторно за допомогою мови Python використовуючи RxPY(реалізація ReactiveX для Python)

# Домашнє завдання №27а

Виконати домашнє завдання №25 повторно(без використання RxJS) за допомогою мови JavaScript для платформи NodeJS(застосовуючи тільки рушій цієї платформи).

*\* код домашнього завдання №25 та домашнього завдання №27 має відрізнятися тільки кількома останніми стрічками, які в першому випадку показують використання реактивної парадигми програмування, а в другому випадку – подійно-орієнтованої*

# Домашнє завдання №28

Виконати домашнє завдання №25 повторно як альтернативну низькорівневу реалізацію домашнього завдання №27 мовою С. Для цього використати poll(системний виклик ОС Linux) з відстеженням POLLIN у revents з структури pollfd при передачі даних за допомогою неіменованого каналу(pipe) до обробника вводу(в прикладі він називається inputHandler). (*Це імітує process.stdin.on( 'keypress', inputHandler)* з прикладу коду [домашнього завдання №27](#)). Саму передачу даних виконує додатковий скануючий обробник(в прикладі він називається stdInInputHandler), який виступає в ролі аналога поведінки вбудованої реалізації process.stdin з NodeJS.

Замість використання системного виклику poll також можна використати системний виклик select, тоді відстежуватися буде безпосередньо неіменований канал.

# Домашнє завдання №276

У 28 домашньому завданні потрібно було виконати домашнє завдання №25 повторно як альтернативну низькорівневу реалізацію домашнього завдання №27 мовою С. Сам рушій платформи NodeJS побудований на основі бібліотеки libuv, яка була створена для заміни libeio(імплементує Tread Pool) та libev(імплементує Event Loop). (*Рушій JavaScript для NodeJS це V8, але рушієм подійно-орієнтованої парадигми у NodeJS є саме libuv*). В якості самостійної роботи пропонується знову виконати домашнє завдання №25 повторно як альтернативну низькорівневу реалізацію домашнього завдання №27 мовою С, але на цей раз за допомогою бібліотеки libuv.



<https://repl.it/languages/cpp>



<https://www.tutorialspoint.com/codingground.htm>

