Лекція 1

ОСНОВИ ТЕХНОЛОГІЇ ПРОГРАМУВАННЯ

Лекція 1. Основи технології програмування

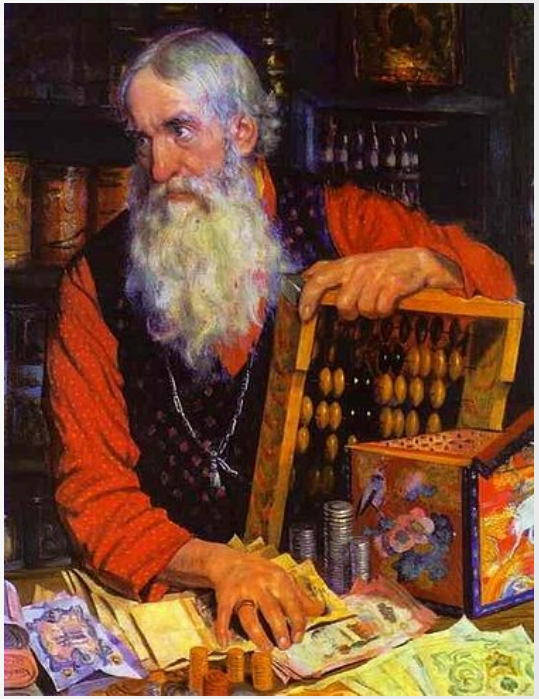
План

1. Історія створення обчислювальної техніки
2. Поняття технології програмування
3. Основні сучасні парадигми та технології програмування
4. Література

1. Історія створення обчислювальної техніки

Абак (лічильна дошка)
З'явилися приблизно у третьому тисячоріччі до н.е. у Стародавньому Вавилоні

Рахівниця
З'явилися як розвиток китайського рахункового пристрою суаньпань приблизно у XIV столітті і активно використовуються досі!!!



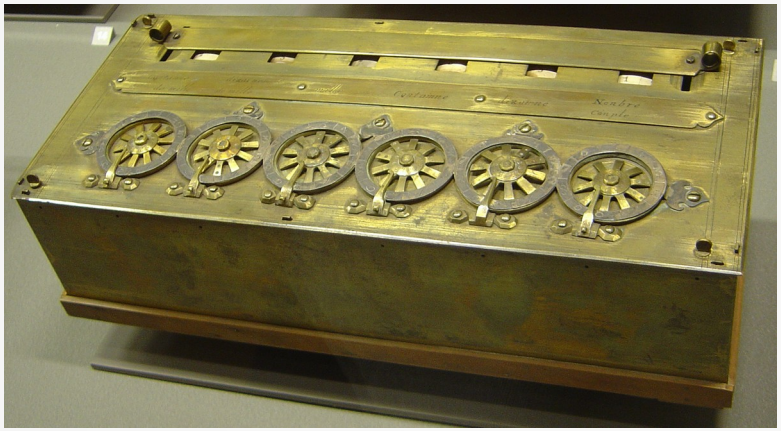
Купець
Б. М. Кустодієв ,1918 р.

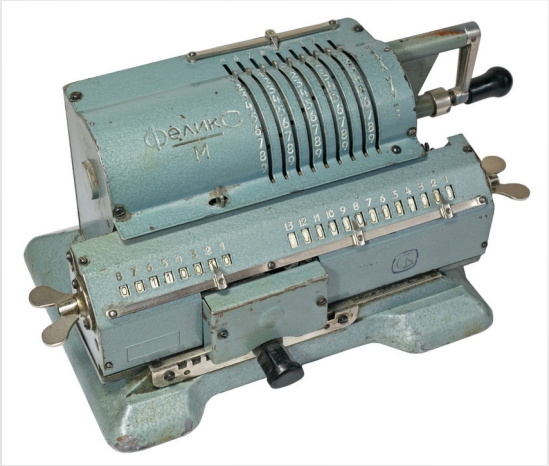
1. Історія створення обчислювальної техніки

Блез Паскаль
(1623-1662)

Паскаліна (калькулятор Паскаля)

Створена французьким ученим Блезом Паскалем у 1642 році

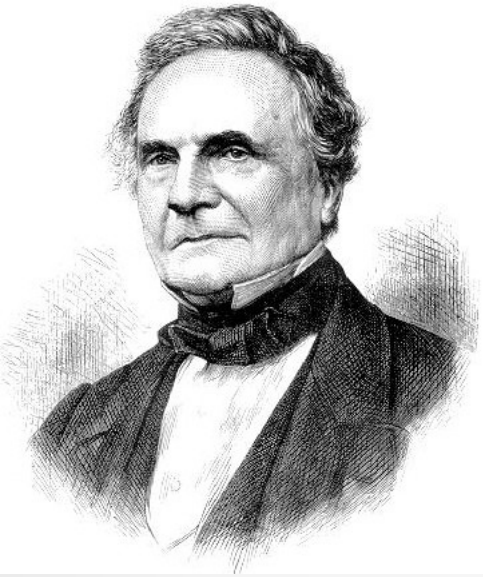




Арифмометр

Портативна механічна обчислювальна машина, призначена для точного множення та ділення. Випускалася у СРСР до 1978 року.

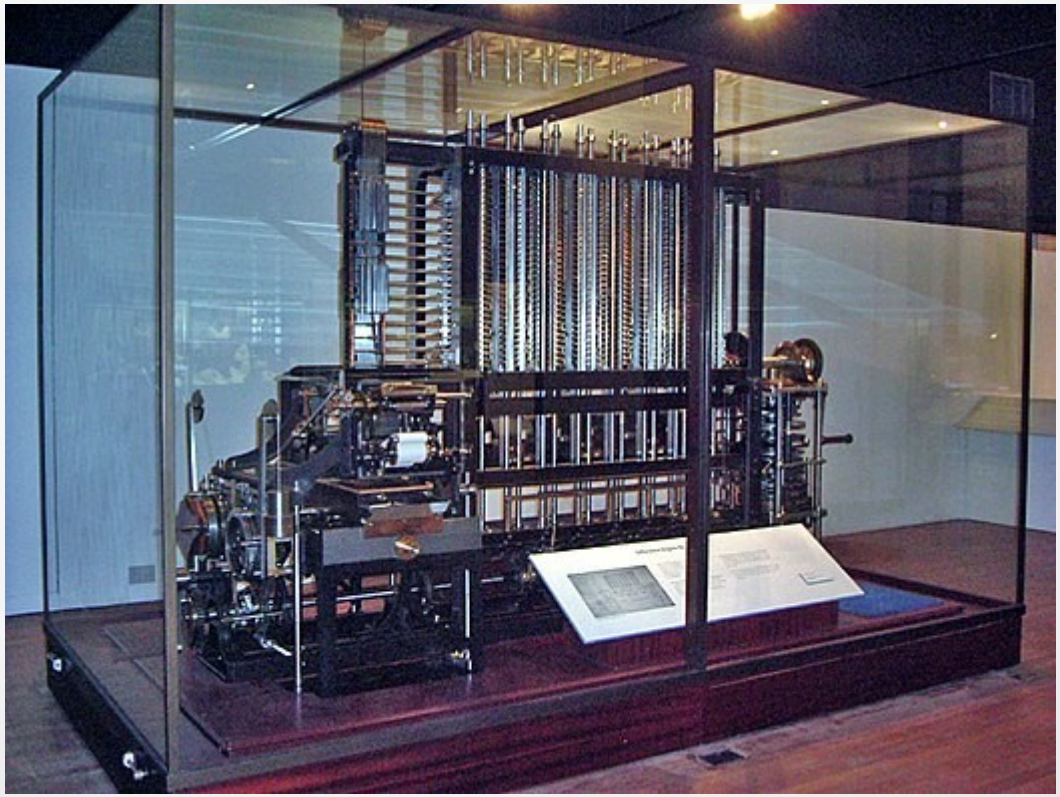
1. Історія створення обчислювальної техніки



Чарльз Беббідж
(1791-1871)

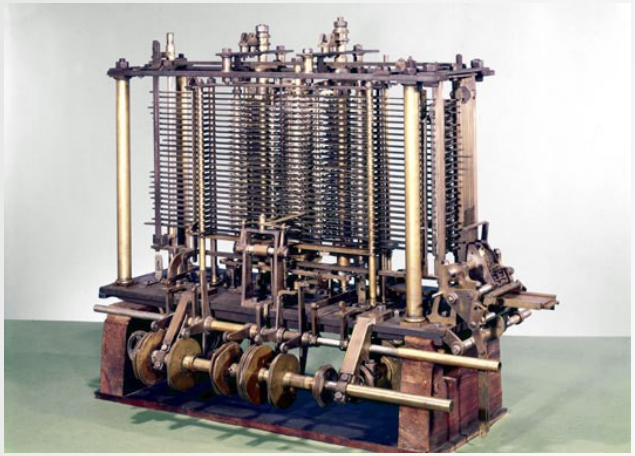
Різницева машина Беббіджа

Автоматичне обчислення значення багаточленів до шостого степеня з точністю до 18-го знака.
Запропонована Ч. Беббіджем у 1822 році



Сучасна реконструкція різницевої машини Беббіджа

1. Історія створення обчислювальної техніки



Аналітична машина Беббіджа

Прообраз сучасного комп'ютера, що складається з арифметичного пристрою («млина»), пам'яті («складу») та пристрою введення-виводу, реалізованого за допомогою перфокарт, які могли бути використані як для введення даних в машину, так і для збереження результатів обчислень, якщо пам'яті було замало.

Ткацький верстат з картами Жаккара

Ада Лавлейс (1815-1852)

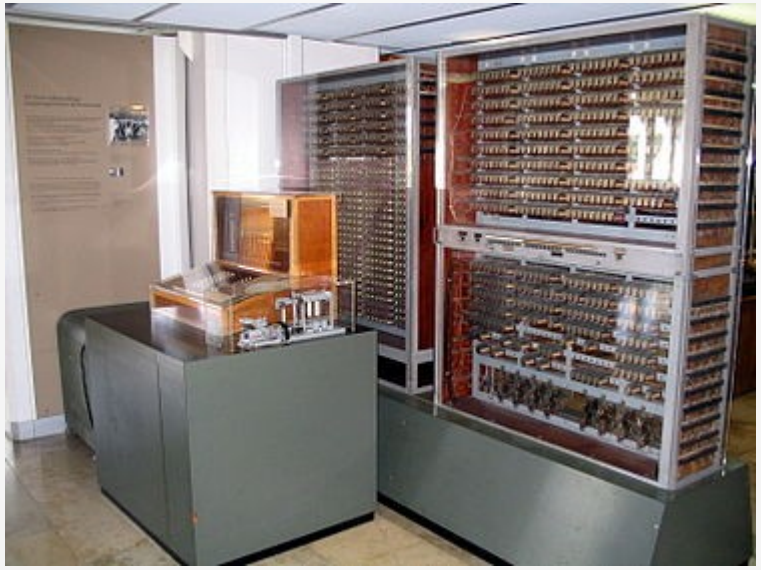
Ада Лавлейс вважається першим програмістом, оскільки вона вперше описала алгоритм обчислення чисел Бернуллі на аналітичній машині

1. Історія створення обчислювальної техніки

Конрад Цузе
(1910-1995)

Z3

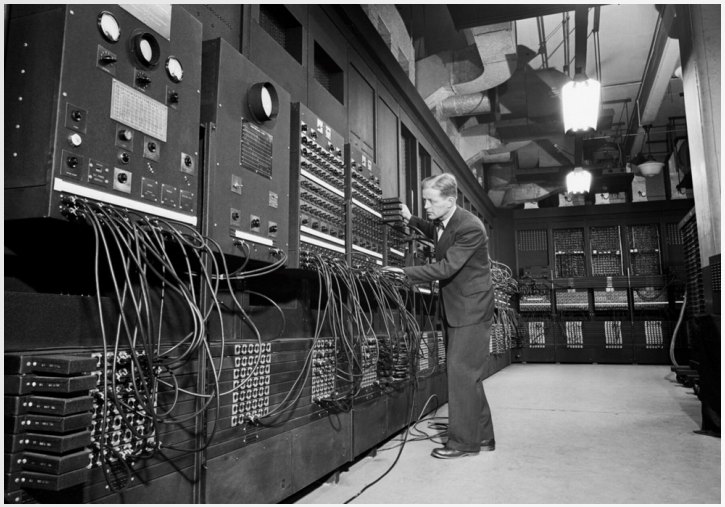
Перша працездатна повнофункціональна програмно керована обчислювальна машина (фактично – перший електромеханічний комп'ютер). Побудована у 1941 році в Німеччині).



Тактова частота: 5,3 Гц.

К. Цузе розробив першу у світі високорівневу мову програмування, яку він назвав **Планкалкюль** (Plankalkül – обчислення планів).

1. Історія створення обчислювальної техніки



ENIAC (Electronic Numerical Integrator and Computer)

Перший електронний цифровий обчислювач загального призначення, який можна було перепрограмувати на вирішення широкого спектра завдань. Побудований у 1945 році у США).

Основні характеристики

- Вага – 30 тон.
- Обчислювальна потужність – 357 операцій множення або 5000 операцій додавання в секунду.
- Тактова частота – 100 кГц (додавання виконувалося за 1 такт, множення – за 14 тактів).

1. Історія створення обчислювальної техніки

LEO

Перший приватний комп'ютер, який використовувався для обробки комерційних даних (використовувався для розрахунку ціни на продукти харчування). Побудований у 1951 році в Англії).

Тактова частота: 500 КГц.

Altair 8800

Перший у світі персональний комп'ютер (мікрокомп'ютер). Розроблений у США у 1975 році.

Тактова частота: 2 МГц

2. Поняття технології програмування

Програма – це комбінація комп'ютерних інструкцій та даних, яка дозволяє апаратному забезпеченню обчислювальної системи виконувати обчислення або функції управління (стандарт ISO/IEC/IEEE 24765:2010)

Програмне забезпечення (ПЗ) – це сукупність програм, призначених для керування комп'ютером.

Класифікація ПЗ за сферою застосування:

- наукове;
- для побутових пристроїв;
- для забезпечення роботи обладнання;
- для бізнесу;
- для організації роботи мережі;
- ...

Класифікація ПЗ за призначенням:

- прикладне;
- системне;
- інструментальне.

Програмування – процес розробки ПЗ, що поєднує в собі елементи науки (математики), інженерії та мистецтва. У вузькому значенні програмування – кодування алгоритму з допомогою деякої мови програмування (алгоритмічного мови). У широкому – тривалий та творчий процес створення програм.

2. Поняття технології програмування

Технологія програмування (ТП) – це послідовність процесів, що виконуються розробниками ПЗ (програмістами) для створення алгоритму вирішення задачі, її кодування мовою програмування, тестування та налагодження, а також передачі замовнику та наступного супроводу.

Іншими словами, ТП – це сукупність засобів і методів розробки ПЗ та порядок їх застосування.

Технологічний ланцюжок програмування зазвичай містить такі основні етапи:
1) постановка задачі;
2) розробка або пошук алгоритму розв’язання задачі;
3) специфікація вихідних даних програми;
4) специфікація функцій програми;
5) проектування програми;
6) кодування мовою програмування;
7) тестування та налагодження програми;
8) тестова експлуатація.

2. Поняття технології програмування

Постановка задачі – поділяється на неформальну та формальну. На неформальному етапі відбувається діалог між замовником та розробником, головним результатом якого є однакове розуміння задачі та результатів її вирішення. За необхідності це фіксується у вигляді формального документа (технічного завдання), де відображено всі вимоги замовника до майбутнього ПЗ.

Розробка або пошук алгоритму – етап, під час якого розробляється алгоритм розв'язання задачі (якщо вона нова) або вибирається оптимальний з вже наявних.

Специфікація вихідних даних програми – етап, на якому формалізуються формати використовуваних даних, способи їх зберігання та обробки тощо.

Специфікація функцій програми – розробка опису функціональності програми.

Проектування програми – етап, на якому створюється детальний опис програми, її функціоналу, структури і т. п. Як правило, для цього використовуються спеціалізовані мови формального або графічного опису, наприклад UML.

2. Поняття технології програмування

Кодування мовою програмування – безпосередньо програмна реалізація алгоритму на вибраній мові (мовах) програмування.

Тестування та налагодження програми – процес пошуку та усунення можливих помилок у ПЗ.

Тестова експлуатація – процес перевірки програми при її експлуатації в реальних умовах з метою виявлення в ній недоліків та помилок з подальшим усуненням.

3. Основні сучасні парадигми та технології програмування

Отже **технологія програмування** – це апробована стратегія створення програм, яка викладається у вигляді певних методик із описами проектних процедур та операцій.

До найпоширеніших сучасних технологій розробки програмного забезпечення належать такі:

1) **процедурне програмування;**
2) **структурне програмування**;
3) **об’єктно-орієнтоване програмування**;
4) **візуальне програмування**.

3. Основні сучасні парадигми та технології програмування

У кожній з цих технологій використовується одна або кілька **парадигм програмування** – сукупностей принципів, методів та понять, що визначають спосіб розробки програм.

Парадигми програмування являють собою різні підходи до написання програм (філософії). Для використання кожної з них необхідні свій тип мислення, особлива школа навчання, прийоми та способи програмування, що визначаються мовою програмування, яка використовується.

Існують чотири основні парадигми, що описують більшість сучасних методів програмування:

- **імперативна**;
- **аплікативна**;
- **заснована на системі правил**;
- **об'єктно-орієнтована**.

3. Основні сучасні парадигми та технології програмування

Імперативна парадигма – найпоширеніший і природніший підхід, що відповідає архітектурі стандартного комп'ютера, який виконує команди послідовно.

Програма імперативною мовою складається з послідовності операторів, що виконуються один за одним:

```
Оператор_1;
Оператор_2;
Оператор_3;
…
```

До імперативних мов належать, наприклад, **Fortran**, **C**, **Pascal**, **Ada** тощо.

3. Основні сучасні парадигми та технології програмування

Приклад програми імперативною мовою програмування BASIC

```
10 CLS 'Очищення екрану
20 PRINT "Hello" 'Заголовок в першому рядку
30 'Цикл, що виводить лінію під заголовком, на всю ширину екрану
40 FOR I=1 TO 80
50 PRINT "=";
60 NEXT I
65 'Введення символьних даних від користувача
70 INPUT "First Name:  ",N$
80 INPUT "Last Name: ",S$
90 INPUT "Patronymic: ",T$
95 'Вирізаємо копію перших символів з імені та по батькові
100 N2$=LEFT$(N$,1)
110 T2$=LEFT$(T$,1)
120 'Виводимо результат
130 PRINT "Your name: ";S$;" ";N2$;". ";T2$;"."
140 INPUT "Repeat program? (Y/N) ",U$
150 IF U$="Y" THEN GOTO 10
160 END
```

3. Основні сучасні парадигми та технології програмування

Аплікативна парадигма – базується на ідеї розгляду функції, яку виконує програма. Іншими словами, яку функцію потрібно застосувати до початкового стану обчислювальної машини, щоб отримати бажаний результат?

Мови, що підтримують таку парадигму, називаються **аплікативними** або **функціональними**. Їхній синтаксис, як правило, виглядає наступним чином:

Функція_n(... функція_2(функція_1(дані))...)

До функціональних відносяться такі мови, як **Haskell**, **Lisp**, **ML** та інші.

3. Основні сучасні парадигми та технології програмування

Приклади програм на аплікативних мовах програмування

Haskell

```
calc :: String -> Float
 calc = head . foldl f [] . words
   where
     f :: [Float] -> String -> [Float]
     f (x:y:zs) "+"    = (y + x):zs
     f (x:y:zs) "-"    = (y - x):zs
     f (x:y:zs) "*"    = (y * x):zs
     f (x:y:zs) "/"    = (y / x):zs
     f (x:y:zs) "FLIP" = y:x:zs
     f (x:zs)   "ABS"  = (abs x):zs
     f xs       y      = read y : xs
```

Lisp

```
(SmartPlus 1 2)
==> 3
(type-of (SmartPlus 1 2 ))
==> (INTEGER 0 461168601842738790 3)
(SmartPlus 1 1.2)
==> 2.2
(type-of (SmartPlus 1 1.2))
==> SINGLE-FLOAT
(SmartPlus 2 2/3)
==> 8/3
(type-of (SmartPlus 2 2/3))
==> RATIO
(SmartPlus "abc" 20)
==> NIL
(type-of (SmartPlus "abc" 20))
==> NULL
```

3. Основні сучасні парадигми та технології програмування

Парадигма, що базується на системі правил (**логічне програмування**). Мови, засновані на цій парадигмі, здійснюють перевірку наявності необхідної умови згідно із заданими логічними правилами і у разі її виявлення виконують відповідну дію.

Оператори у програмі при використання цієї парадигми виконуються не у тій послідовності, у якій визначено у програмі. Порядок виконання визначають логічні умови. Синтаксис таких мов виглядає так:

умова_1 –> дія_1 умова_2 –> дія_2 ...
умова_n -> дія_n

Найбільш відомою логічною мовою програмування є **Prolog**.

3. Основні сучасні парадигми та технології програмування

Приклад програми на мові логічною програмування Prolog

```
parent("Tom","Jake").
parent("Janna","Jake").
parent("Tom","Tim").
male("Tom").
male("Tim").
male("Jake").
female("Janna").

brother(X,Y):-
parent(Z,X),parent(Z,Y),male(X),male(Y),X\=Y.
```

3. Основні сучасні парадигми та технології програмування

Об'єктно-орієнтована парадигма. У цій моделі будуються складні структури (типи) даних, для операцій над якими вводяться певні набори методів. Створювані об’єкти можуть успадковувати характеристики інших (більш простих) об'єктів. Завдяки такій можливості об’єктно-орієнтовані програми мають високу ефективність, властиву програмам, написаним імперативними мовами. Можливість розробки різних класів, які використовують обмежений набір об’єктів даних, обумовлює гнучкість та надійність, які властиві аплікативним мовам.

До об'єктно-орієнтованих мов відносяться **Java**, **C#**, **Python**, **C++** (не повною мірою) та багато інших.

3. Основні сучасні парадигми та технології програмування

Приклад програми об'єктно-орієнтованою мовою C#

```
// assembly: System.dll
// assembly: System.Drawing.dll
// assembly: System.Windows.Forms.dll
using System;
using System.Drawing;
using System.Windows.Forms;

namespace WindowsForms
{
    public class Program
    {
        [STAThread]
        public static void Main()
        {
            new DemoForm().ShowDialog();
        }
    }
```

```
public class DemoForm : Form
    {
        Label label = new Label();

        public DemoForm()
        {
            label.Text = "Hello World!";
            this.Controls.Add(label);
            this.StartPosition =
                    FormStartPosition.CenterScreen;
            this.BackColor = Color.White;
            this.FormBorderStyle =
                    FormBorderStyle.Fixed3D;
        }
    }
}
```

3. Основні сучасні парадигми та технології програмування

На ранніх етапах розвитку програмування програми писалися у вигляді прямої послідовності машинних команд (машинного коду) і будь-які технології програмування були відсутні.

Приклад програми “Hello, world!” для процесора архітектури x86 (у шістнадцятковому форматі):

BB 11 01 B9 0D 00 B4 0E 8A 07 43 CD 10 E2 F9 CD 20 48 65 6C 6C 6F 2C 20 57 6F 72 6C 64 21

Пізніше для розробки програмного забезпечення почали використовуватися машинно-залежні низькорівневі мови програмування – **асемблери**.
Приклад програми “Hello, world!” на мові асемблера для операційної системи Linux:

```
ection     .text
global      _start                          ;must be declared for linker (ld)
_start:                                     ;tell linker entry point
    mov     edx,len                         ;message length
    mov     ecx,msg                         ;message to write
    mov     ebx,1                           ;file descriptor (stdout)
    mov     eax,4                           ;system call number (sys_write)
    int     0x80                            ;call kernel
    mov     eax,1                           ;system call number (sys_exit)
    int     0x80                            ;call kernel
section     .data
msg     db  'Hello, world!',0xa             ;our dear string
len     equ $ - msg                         ;length of our dear string
```

3. Основні сучасні парадигми та технології програмування

Зі збільшенням розмірів програм стали виділяти їх відокремлені частини та оформляти їх як **підпрограми**. Частина таких підпрограм об’єднувалася в **бібліотеки**, з яких підпрограми можна було включати до робочих програм і потім викликати з робочих програм. Це започаткувало **процедурне програмування** – велика програма представлялася сукупністю процедур-підпрограм. Одна з підпрограм була головною і з неї розпочиналося виконання програми.

Процедурний підхід вимагає структурування майбутньої програми, поділу її на окремі процедури. При розробці окремої підпрограми про інші процедури потрібно знати тільки їх призначення та спосіб виклику. З’явилася можливість переробляти окремі процедури, не торкаючись решти програми, скорочуючи при цьому витрати праці та машинного часу на розробку та модернізацію програм.

До перших мов, що підтримують процедурний підхід, відносяться **Fortran**, **Algol**, **Cobol** та інші.

3. Основні сучасні парадигми та технології програмування

Наступним кроком у поглибленні структурування програм стало **структурне програмування**, у якому програма розглядалися як послідовність канонічних структур: **лінійних ділянок**, які складалися з операторів, що виконуються послідовно, а також **циклів** і **розгалужень**.

Завдяки відмові від використання оператора безумовного переходу (**goto**) та використання спеціальних стилів форматування з'явилася можливість читати та перевіряти вихідний текст програми як послідовний текст, що підвищило продуктивність праці програмістів при розробці, налагодженні та супроводі програм.

До мов, що підтримують структурне програмування, відносяться **C**, **C++**, **Pascal**, **Python** тощо.

3. Основні сучасні парадигми та технології програмування

Всі **універсальні мови програмуванн**я, незважаючи на відмінності в синтаксисі і ключових словах, що використовуються, реалізують одні й ті ж канонічні структури: оператори присвоювання, цикли та розгалуження. У всіх сучасних мовaї є базові типи даних (цілі та дійсні арифметичні типи, рядкові і т.п.), є можливість використання агрегатів даних, у тому числі масивів та структур (записів).

Однак, при розробці програмного забезпечення для вирішення конкретної прикладної задачі бажано наявність більшої концептуальної близькості тексту програми до опису задачі. Цього можна досягти двома основними способами:
1) створенням мови програмування (**мови-оболонки**), що містить якомога більше типів даних, та вибором для кожного класу завдань деякої підмножини цієї мови (наприклад, **PL/1**);
2) побудовою мови, що розширюється (**мови-ядра**), що містить невелике ядро, що допускає можливість розширення і доповнює мову типами даних і операторами, які відображають концептуальну сутність конкретного класу завдань (**C++**).

3. Основні сучасні парадигми та технології програмування

Об’єктно-орієнтований підхід (ООП) до програмування базується на таких основних ідеях:

- програма є моделлю деякого реального процесу або об’єкту (частини реального світу);
- модель реального світу або його частини може бути описана як сукупність об’єктів, що взаємодіють між собою;
- об’єкт описується набором параметрів, значення яких визначають стан об’єкта, та набором операцій (дій), які може виконувати об’єкт;
- взаємодія між об’єктами здійснюється посилкою спеціальних повідомлень від одного об’єкта до іншого, при цьому повідомлення, отримане об’єктом, може вимагати виконання певних дій, наприклад зміни стану об’єкта;
- об’єкти, описані одним і тим же набором параметрів і здатні виконувати однаковий набір дій, належать до одного класу однотипних об’єктів.

ООП передбачає, що при розробці програми 1) мають бути визначені класи об’єктів, що використовуються у програмі; 2) побудовані їх описи; 3) створено екземпляри необхідних об’єктів і визначено взаємодію між ними.

До мов, що підтримують ООП, відносяться **C++**, **C#**, **Object Pascal**, **Python** та інші.

3. Основні сучасні парадигми та технології програмування

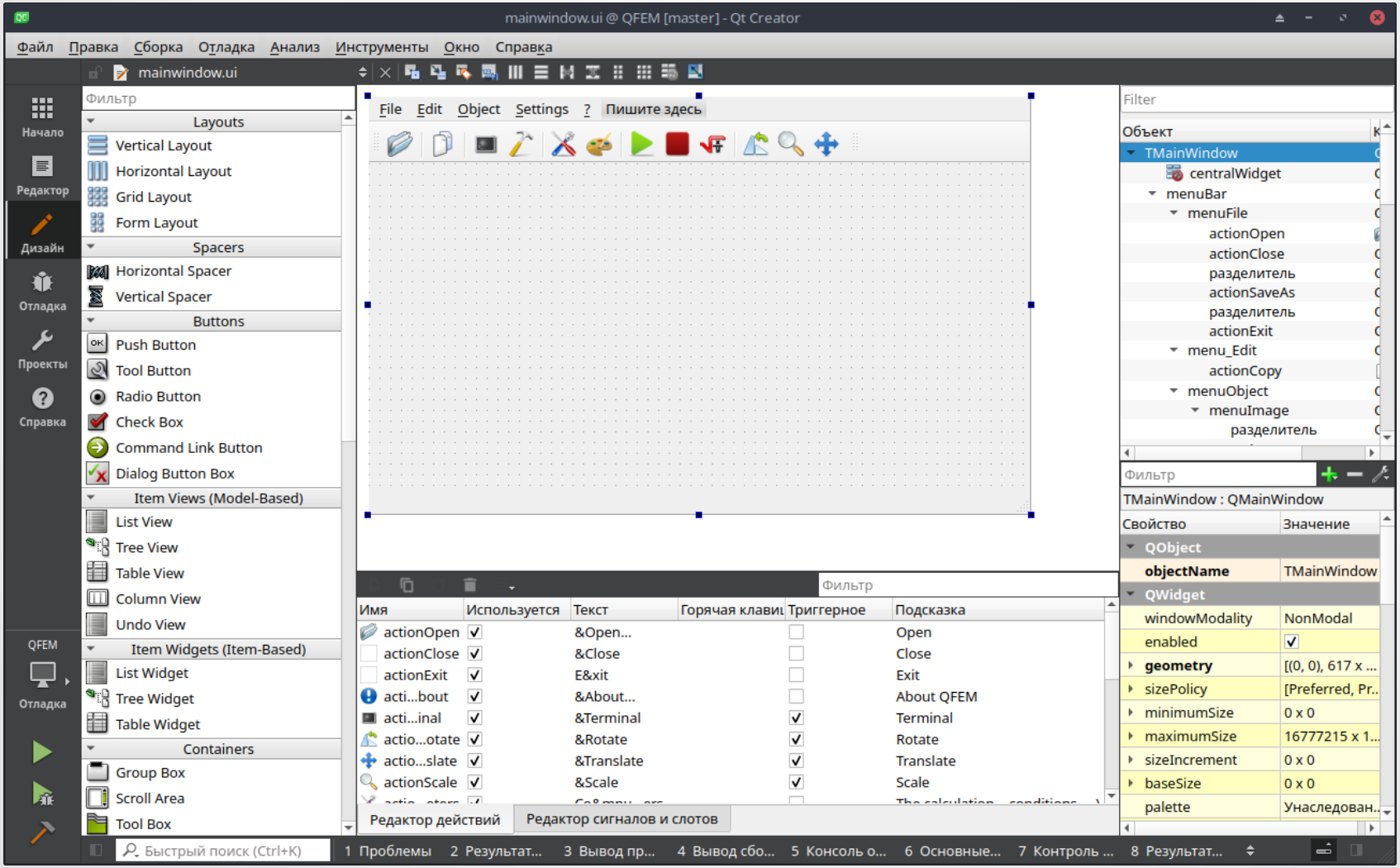
Візуальне програмування є в наш час однієї з найпопулярніших технологій розробки ПЗ. Головною його ідеєю є автоматизація з використанням спеціального інструментарію розробки графічного інтерфейсу користувача (**GUI** – Graphical User Interface) для програм, що працюють у середовищі ОС із графічним інтерфейсом (наприклад, **Windows** або **Linux**).

На початковому етапі візуального програмування, як правило, створюються екранні форми, що відповідають головному та дочірнім вікнам програми, а також визначаються їх властивості та поведінка. Потім з використанням об'єктно-орієнтованого програмування реалізується функціональна складова програми (її бізнес-логіка).

До найвідоміших середовищ візуального програмування відносяться такі комерційні системи, як **Microsoft Visual Studio**, **Embarcadero RAD Studio**, безкоштовні **Qt Creator**, **Eclipse** та інші.

3. Основні сучасні парадигми та технології програмування

Приклад використання середовища Qt Designer для розробки програми GUI



4. Література

1. Камаев В.А. Технологии программирования: Учебник / В.А. Камаев, В.В. Костерин. – 2-е изд., перераб. и доп. – М.: Высш. шк., 2006. – 454 с.
2. Прата С. Язык программирования С: Лекции и упражнения. – М.: Вильямс, 2006. – 960 с.
3. Страуструп Б. Язык программирования С++. Специальное издание. Пер. с англ. – М. : Издательство Бином, 2011. – 1136 с.
4. Скотт М. Эффективный и современный С++: 42 рекомендации по использованию С++ 11 и С++14.: Пер. с англ. – М.: ООО "ИЛ. Вильяме", 2016. – 304 с.
5. Шлее М. Qt 4.5. Профессиональное программирование на С++. – Спб. : БХВ-Петербург, 2010. – 896 с.