UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN
The Faculty of Technical Sciences
POLAND, 10-957 Olsztyn, M. Oczapowskiego 11
tel.: (48)(89) 5-23-32-40, fax: (48)(89) 5-23-32-55
URL: http://www.uwm.edu.pl/edu/sobieski/ (in Polish)

Języki Programowania

Generacje języków programowania

Wojciech Sobieski

Olsztyn, 2001-2021

Historia komputerów

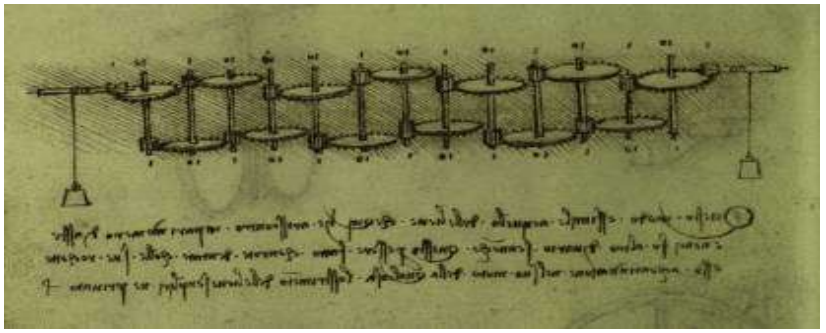
Starożytność – liczenie na czarnych i białych kamieniach, liczydła (Soroban, Abacus).

Abacus

Soroban

Historia komputerów

Leonardo da Vinci (1452-1519) – 13 marca 1967 roku amerykańscy naukowcy pracujący w Madrycie w Bibliotece Narodowej Hiszpanii napotkali dwie nieznane dotąd prace Leonarda da Vinci nazwane „Codex Madrid" dotyczące maszyny liczącej. Dr Roberto Guatelli znany ekspert w dziedzinie twórczości Leonarda, w roku 1968 odtworzył tą maszynę (obecnie nie wiadomo gdzie ona się znajduje).



maszyna Leonarda da Vinci – projekt i współczesna replika

Historia komputerów

Na początku XVII wieku **John Neper** (1550-1617) opublikował najpierw swoje dzieło o logarytmach a następnie przedstawił system wspomagający wykonywanie mnożenia, zwany pałeczkami Nepera. Genialność tego systemu polegała na sprowadzeniu mnożenia do serii dodawań. Pomysł Nepera wykorzystało wielu konstruktorów urządzeń liczących, jemu współczesnych i żyjących po nim.

pałeczki Neppera

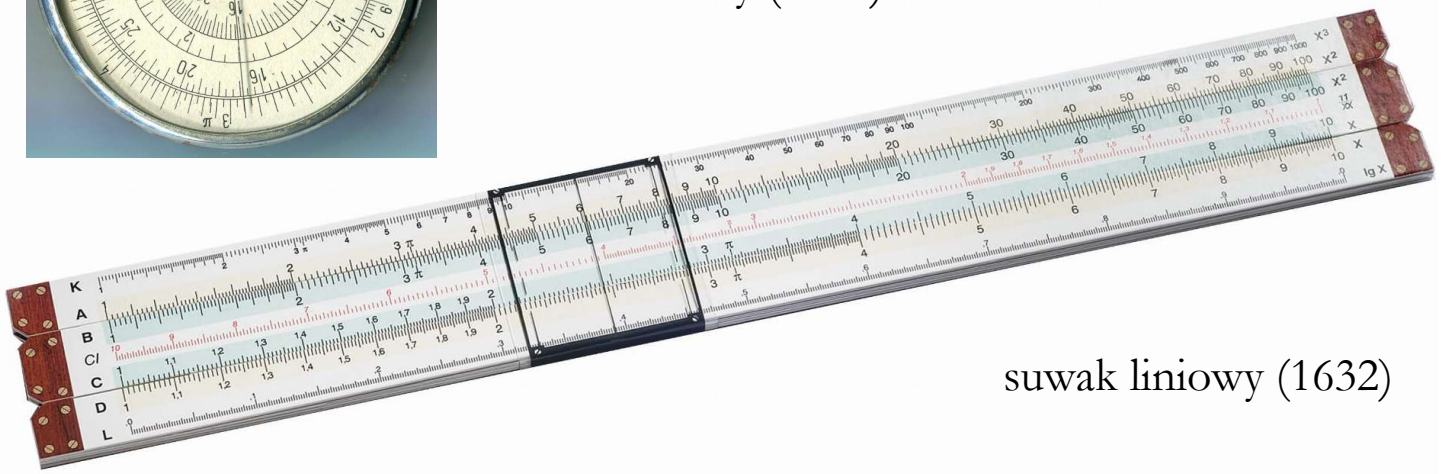
Historia komputerów

Konsekwencją dzieła o logarytmach było wynalezienie w roku 1632 przez **Williama Oughtreda** przyrządu obliczeniowego zwanego suwakiem logarytmicznym. Suwaki logarytmiczne wykorzystywane były aż do lat 80-tych XX wieku.

$$\log_a(b\cdot c)=\log_a(b)+\log_a(c)$$

$$\log_a\left(\frac{b}{c}\right)=\log_a(b)-\log_a(c)$$

suwak kołowy (1633)



suwak liniowy (1632)

Historia komputerów

Za twórcę pierwszej w historii mechanicznej maszyny do liczenia jest uznawany **Wilhelm Schickard** (1592-1635), który przez długie lata był zupełnie zapomniany. Schickard opisał projekt swojej czterodziałaniowej maszyny, wykorzystując udoskonalone pałeczki Nepera w postaci walców, w liście do Keplera, któremu miała ona pomóc w jego „astronomicznych" rachunkach. Niestety jedyny zbudowany egzemplarz maszyny spłonął w niewyjaśnionych okolicznościach, a dzisiejsze jej repliki zostały odtworzone dopiero niedawno na podstawie opisu z listu Keplera.

maszyna Schickarda

Historia komputerów

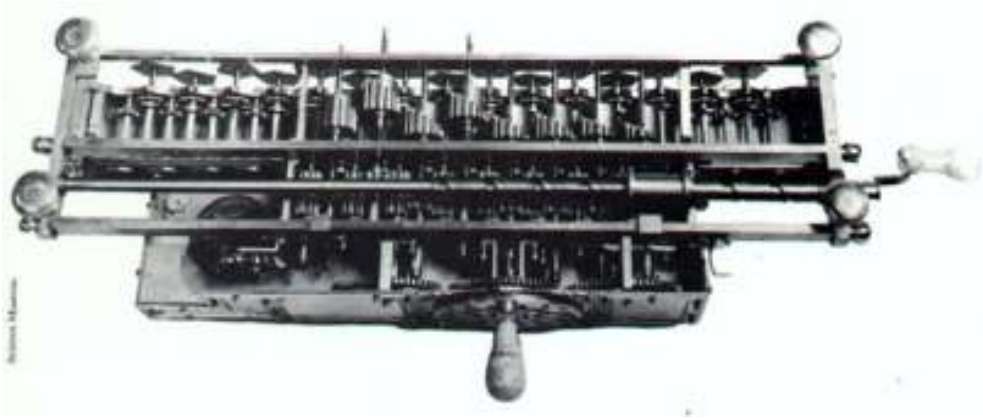
Blaise Pascal (1623-1662) zbudował Pascalinę, maszynę pomagającą w pracy ojcu Pascala, który był poborcą podatkowym. Część maszyn (około 50) była przeznaczona do obliczeń w różnych systemach monetarnych, a część dla różnych miar odległości i powierzchni (z przeznaczeniem dla geodetów). Pascalina wykonywała tylko dwa działania (dodawanie i odejmowanie). Schickard i Pascal wprowadzili w swoich maszynach mechanizm do przenoszenia cyfr przy dodawaniu i odejmowaniu. Obie maszyny miały także pewne możliwości zapamiętywania niektórych wyników pośrednich.

Pascalina

Historia komputerów

Gottfried Wilhelm Leibniz (1646-1716) – odkrył na nowo pochodzący ze starożytnych Chin system dwójkowy (zwany także binarnym) do zapisu liczb. Przypisuje się jemu także zbudowanie pierwszej mechanicznej maszyny mnożącej. Chociaż w tym czasie istniała już Pascalina i Leibniz miał możność zapoznania się z nią w Paryżu, projekt swojej „żywej ławy do liczenia" opisał przed pierwszą wizytą w Paryżu. W maszynie tej wprowadził wiele części, które zostały użyte w późniejszych maszynach biurowych.

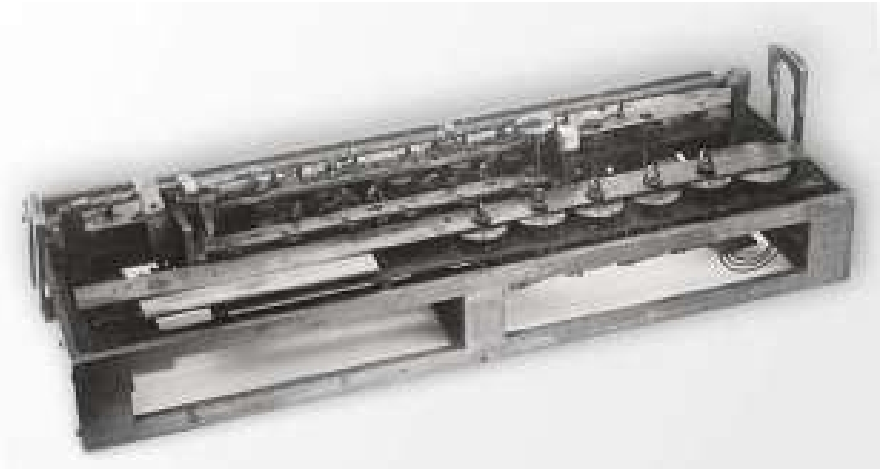
maszyna Leibnitza



Historia komputerów

Abraham Stern (1769-1842), z zawodu zegarmistrz, wykonał serię maszyn, które poza czterema działaniami podstawowymi, wyciągały także pierwiastki kwadratowe. Jedna z jego maszyn, raz uruchomiona, potrafiła wykonać za pomocą mechanizmu zegarowego wszystkie operacje bez ingerencji człowieka. Maszyny skonstruowane przez Sterna okazały się jednak mało praktyczne ze względu na wyjątkowo delikatną budowę.

maszyna Sterma



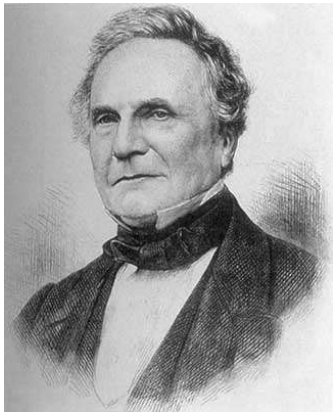
Historia komputerów

Joseph-Marie Jacquard (1752-1834) – ukoronował w 1805 r. kilka wieków rozwoju urządzeń z kodem sterującym procesami (pozytywki itp.), konstruując we Francji krosna, w których kod na taśmie perforowanej sterował haczykami wybierającymi nici odpowiedniego koloru do wzorów na tkaninach. Pomysł ten inspirował Babbage'a i Holleritha, a jego wpływ sięgał aż po von Neumanna.

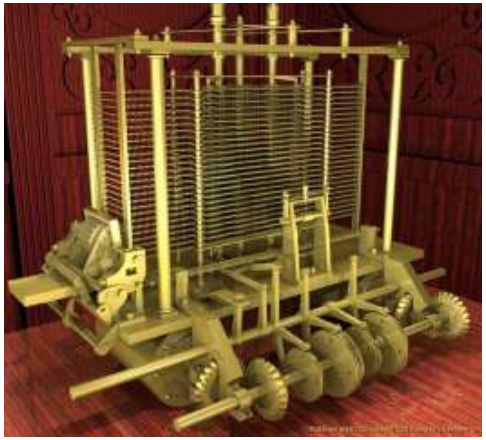
krosna Jacquarda

Historia komputerów

Charles Babbage (1791-1871) – uważany za najwybitniejszego twórcę maszyn liczących, żyjącego przed erą elektroniczną. W odróżnieniu od maszyn Leibniza i Pascala, po ręcznym ustawieniu początkowego stanu, dalsze działania maszyny różnicowej nie wymagały już żadnej ingerencji użytkownika poza kręceniem korbą. Babbage rozdzielił pamięć (zwaną magazynem) od jednostki liczącej (młyna), a całość sterowana była dodatkowym kontrolerem.



maszyny Babbagea



Historia komputerów

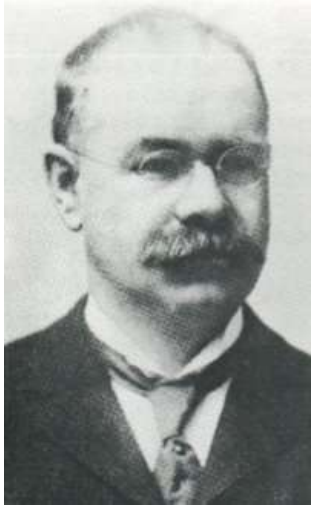
Opis działania maszyny Babbagea trafił w ręce **Ady Augusty hrabiny Lovelace**, znanej w owych czasach z błyskotliwego umysłu. Urzeczona doskonałością projektu uważała, że... „maszyna analityczna tkać będzie wzory algebraiczne, tak jak krosna Jacquarda tkają liście i kwiaty...”. Nie czekając na skonstruowanie maszyny (czego i tak by nie doczekała), Ada zajęła się sporządzaniem opisów jej używania do rozwiązywania konkretnych zadań obliczeniowych. Opisy te nazwano by dzisiaj programami, dlatego uważa się ją za pierwszą programistkę komputerów.

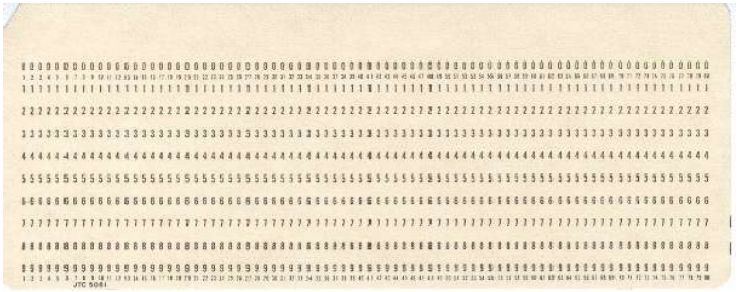
Historia komputerów

George Boole (1815-1864) – matematyk z uniwersytetu w Cork (Irlandia), choć nie skonstruował żadnej maszyny, ma unikalny wkład w konstrukcję bramek logicznych komputera, które są budowane według praw stworzonej przezeń algebry, zwanej algebrą Boole'a. Ta sama algebra zapoczątkowała w połowie XIX w. logikę matematyczną, dostarczającą teoretycznych podstaw informatyki (zagadnienia obliczalności itp.) i metod automatycznego dowodzenia twierdzeń.

Historia komputerów

Herman Hollerith (1860-1929) – jako pierwszy sięgnął po elektryczność, jako źródło impulsów i energii maszyny liczącej. Rozwinął także postać karty perforowanej, na której zapisywano dane i zbudował elektryczny czytnik – sorter kart. Olbrzymim sukcesem Holleritha okazał się spis ludności w Stanach Zjednoczonych w 1890 roku, którego wyniki zostały całkowicie opracowane za pomocą jego urządzeń na podstawie danych zebranych na jego kartach.





maszyna Holeritha oraz używana w niej karta perforowana

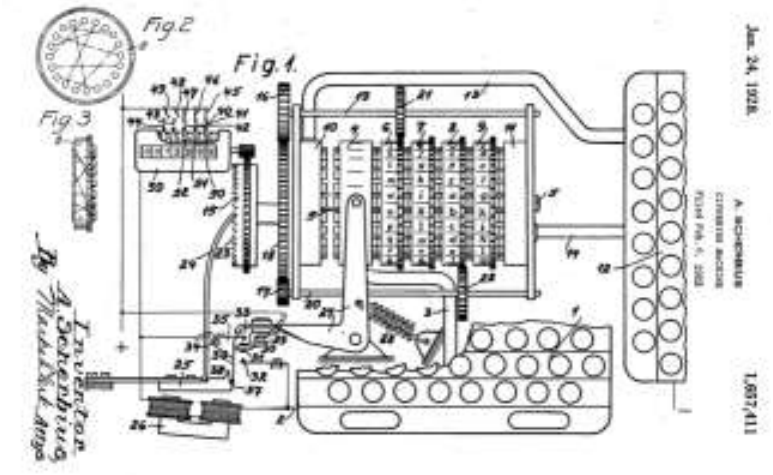
Historia komputerów

Alan Turing (1912-1954) – opublikował 1936 roku przełomową pracę dotyczącą teorii maszyn obliczeniowych i algorytmów; był również konstruktorem popularnych w owym czasie maszyn liczących. Turing sformułował tezę, że na maszynach jego pomysłu można zrealizować każdy algorytm. Do dzisiaj nie obalono tej tezy. Zauważa się, że w związku z tym można przyjąć, iż algorytmem jest dowolny opis wykonania obliczeń na maszynie Turinga.

Test Turinga – test sprawdzający stopień inteligencji maszyn: jeżeli człowiek nie jest w stanie określić, czy rozmawia z innym człowiekiem czy też z maszyną, to taka maszyna zdaje test Turinga.

Historia komputerów

W roku 1919 **Artur Scherbius** otrzymał patent na przenośną, elektromechaniczną maszynę szyfrującą zwaną Enigma (z gr. zagadka). Enigma wykorzystywała zespół obracających się wirników.

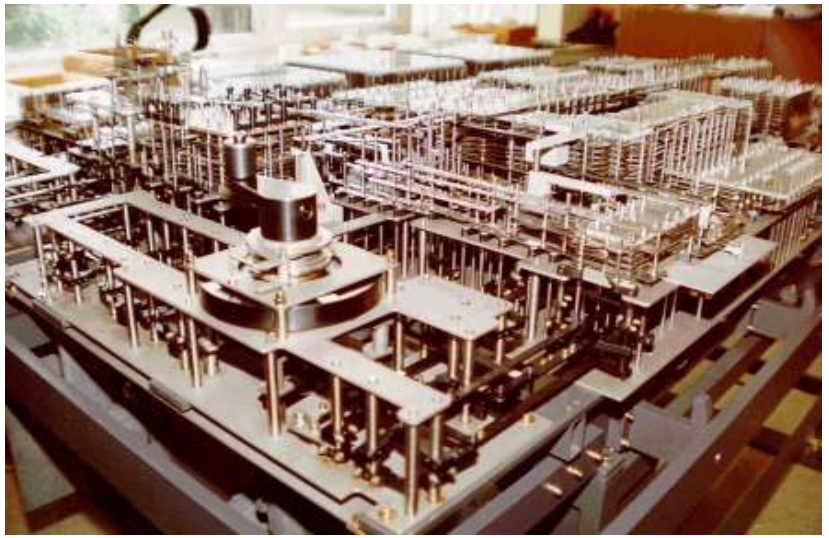


Enigma

Historia komputerów

W 1941 roku **Konrad Zuse** (1910-1995) ukończył w Niemczech prace nad maszyną Z3, która wykonywała obliczenia na liczbach binarnych zapisanych w reprezentacji, nazywanej dzisiaj zmiennopozycyjną, sterowane programem zewnętrznym podawanym za pomocą perforowanej taśmy filmowej. Maszyna Z3 została całkowicie zniszczona w czasie bombardowania w 1945 roku. Następny model maszyny Zusego, Z4 przetrwał i działał do końca lat pięćdziesiątych. Maszyny Zusego były kalkulatorami przekaźnikowymi.

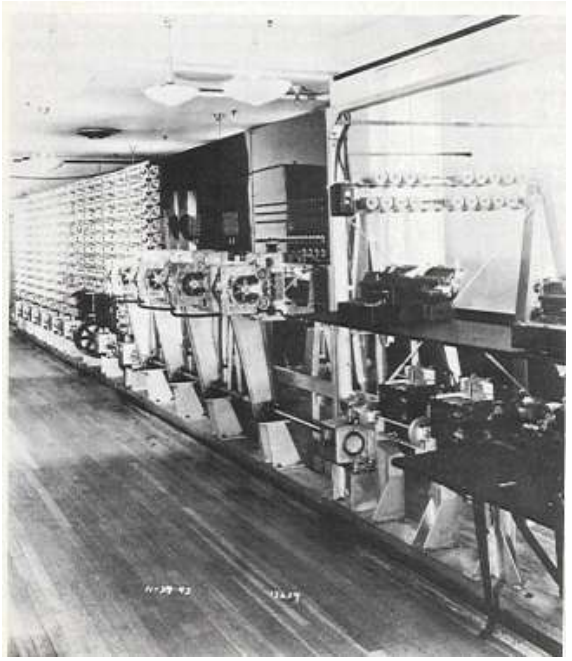
maszyna Z1

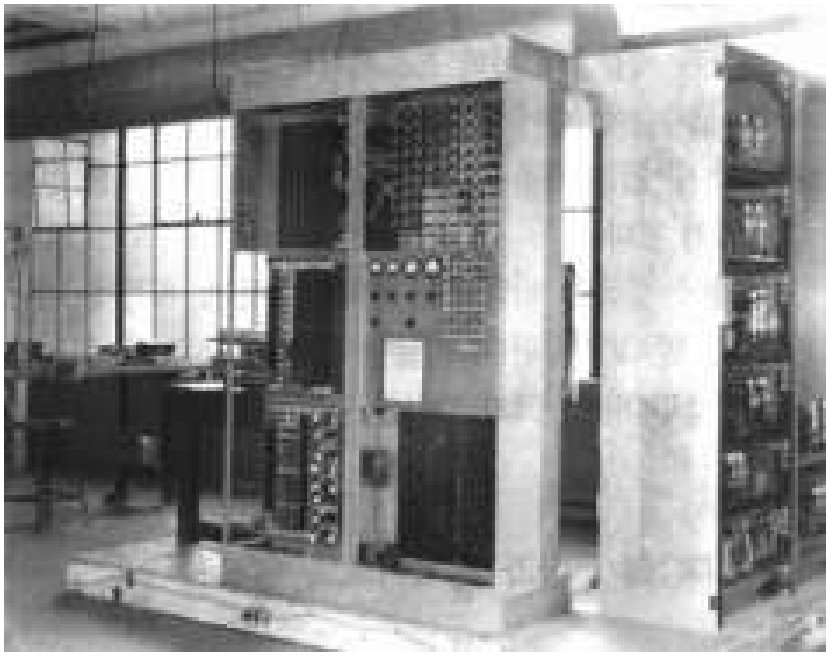


Historia komputerów

1944 – Inżynierowie z Harvardu budują komputer **Mark I** (z lewej).

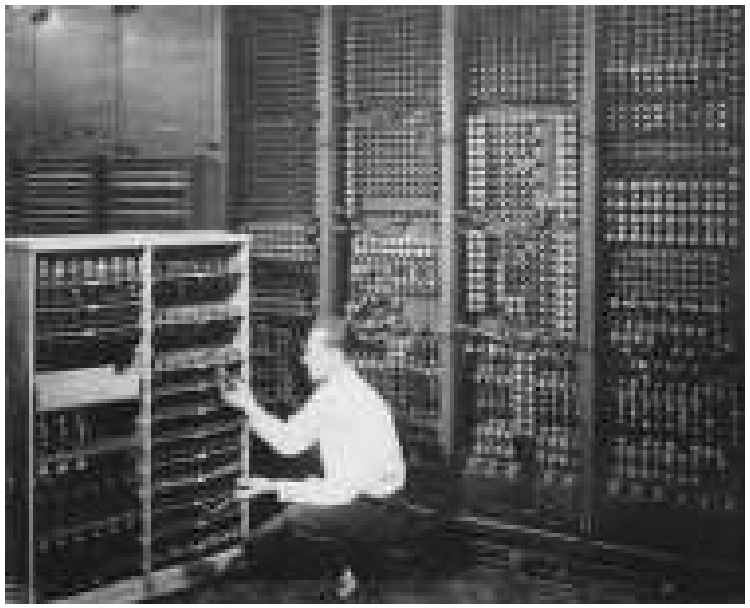
1945 – J.P. Eckert i J. Mauchly budują komputer **EDVAC** (z prawej).





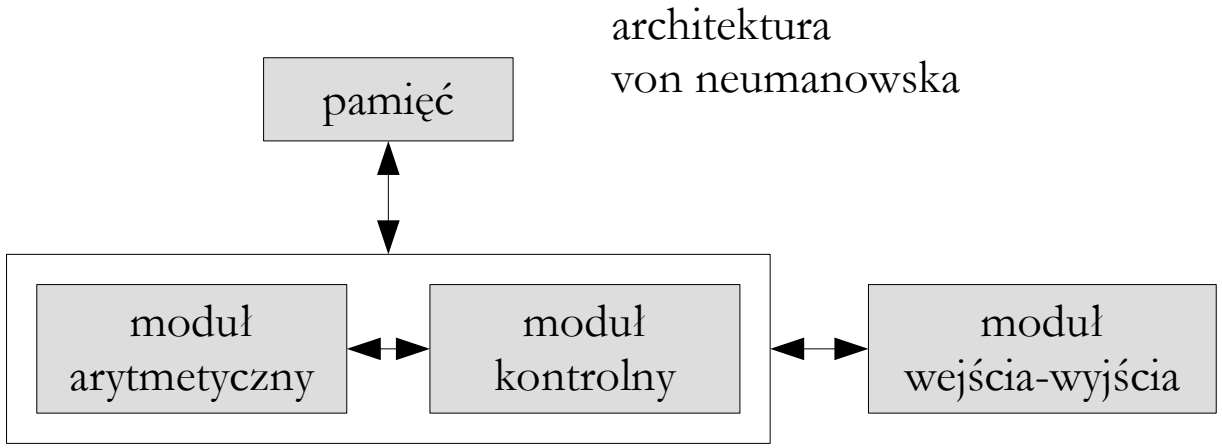
Historia komputerów

W latach 1943-45 zespół specjalistów pod kierunkiem **J.W. Mauchly'ego** i **J.P. Eckerta** zaprojektował i zbudował maszynę ENIAC (*Electronic Numerical Integrator And Computer*). Pierwsze obliczania maszyna ta wykonała w listopadzie 1945 roku. Maszyna ENIAC jest uznawana powszechnie za pierwszy kalkulator elektroniczny, chociaż w 1976 roku okazało się, że wcześniej zaczęły pracować w Wielkiej Brytanii maszyny Coloss I i II. Maszyna ENIAC była konstrukcją złożoną z 50 szaf o wysokości 3 metrów zawierających około 20 tysięcy lamp.



Historia komputerów

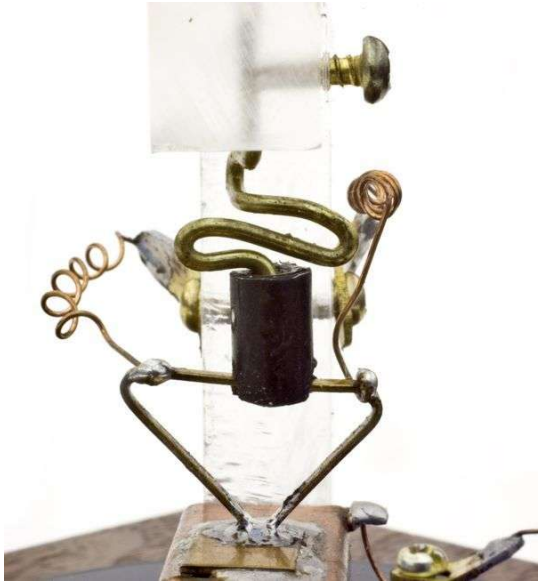
John von Neumann (1903-1957) – z pochodzenia Węgier, był w swoich czasach jednym z najwybitniejszych matematyków. W 1946 roku zainspirował on prace w projekcie EDVAC (*Electronic Discrete Variable Automatic Computer*), których celem było zbudowanie komputera bez wad poprzednich konstrukcji. Zaproponowano architekturę, zwaną odtąd von neumannowską, według której buduje się komputery do dzisiaj.



architektura
von neumanowska

Historia komputerów

1947 – W.B. Shockley, J. Bardeenem i W.H. Brattainem wynajdują tranzystor, za co otrzymują Nagrodę Nobla.



1948 – John von Neumann proponuje architekturę komputerów stosowaną praktycznie do dnia dzisiejszego.

1948 – R. Hamming opracowuje sposób wykrywania błędów w programach.

1949 – J. Mauchly tworzy język programowania **Short Order Code**.

1955 – Bell Telephone Labs produkuje pierwszy komputer oparty na tranzystorach.

Historia komputerów

1956 – IBM opracowuje pierwszy twardy dysk, nazywany **RAMAC**.

1955 – Programiści IBM tworzą język programowania **FORTRAN**.

1958 – J. Kolby opracowuje dla Texas Instruments pierwszy układ scalony.

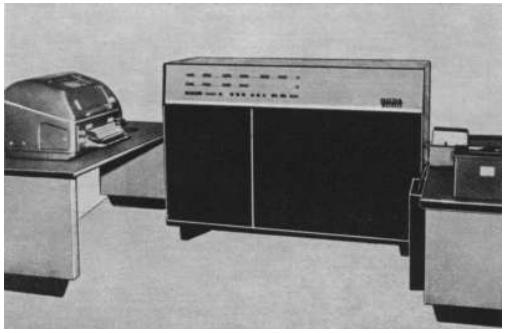
1959 – G.M. Hopper i C. Phillips tworzą język programowania **COBOL**.

1960 – Powstaje język programowania **Algol 60**.

Historia komputerów

1960 – powstaje **PDP-1**, pierwszy komputer wyposażony w monitor i klawiaturę (z lewej).

1963 – Rozpoczęto seryjne wytwarzanie komputera **Odra 1003** (z prawej).



1965 – Digital Equipment Corporation buduje pierwszy minikomputer.

1965 – Zostaje stworzony uproszczony język programowania **BASIC**.

1967 – Texas Instruments produkuje na potrzeby armii pierwszy komputer oparty na układach scalonych z pamięcią półprzewodnikową.

Historia komputerów

1967 – O.J. Dahl i K. Nygaard z Norwegian Computing Centre opracowują język **Simula –** pierwszy obiektowo zorientowany język programowania.

1969 – Departament Obrony USA zleca utworzenie sieci **ARPA-NET**, która połączyła uniwersytety UCLA, UC w Santa Barbara, SRI i University of Utah.

1970 – D. Ritchie i K. Thomson opracowują system operacyjny **Unix** w firmie Bell Labs.

1971 – N. Wirth opracowuje język programowania **Pascal**.

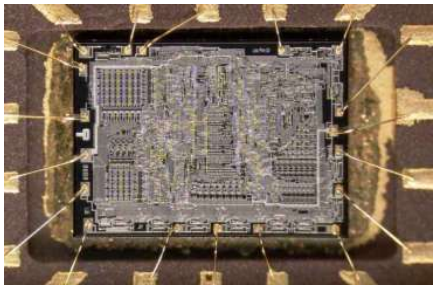
1971 – R. Tomlinson z firmy Bolt Beranek and and Newman wysyła pierwszy **e- mail**.

1972 – D. Ritchie opracowuje język programowania **C** w firmie Bell Labs.

Historia komputerów

1972 – W laboratoriach PARC powstaje obiektowy języka programowania **Smalltalk**, stworzony przez Alana Kaya.

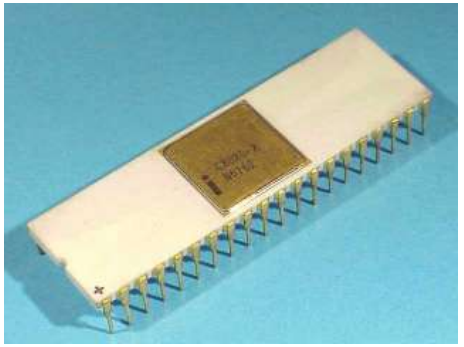
1972 – Powstaje procesor Intel **8008** (200 KhH), pierwszy 8 bitowy układ.



1972 – Powstaje dyskietka 5 1/2 cala.

1973 – Naukowcy z Xerox PARC opracowują eksperymentalny komputer PC. Używa on myszy, sieci Ethernet i GUI.

1974 – W kwietniu powstaje 8 bitowy procesor **8080** (2 Mhz).



Historia komputerów

1975 – Powstaje pierwszy PC - **Altair 8800**.

1975 – W kwietniu Bill Gates i Paul Allen zakładają firmę **Micro-Soft**.

1976 – Steve Jobs i Steve Wozniak budują komputer **Apple I**.

Historia komputerów

1976 – G. Kildall pisze system operacyjny **CP/M** działający na 8-bitowych komputerach z procesorem Intel 8080.

1978 – Powstaje **Wordstar**, pierwszy procesor tekstu.

1978 – W grudniu firma Atari wypuściła komputery
Atari 400 i **800**
z procesorem 6502.

Historia komputerów

1979 – D. Bricklin i B. Franston piszą pierwszy arkusz kalkulacyjny **VisiCalc**.

1979 – Powstaje język programowania **Ada**.

1979 – Sinclair Research przedstawia komputer **ZX80**, wykorzystujący 8-bitowy procesor NEC 3.25 Mhz i mający 1 Mb RAM oraz 4KB ROM.

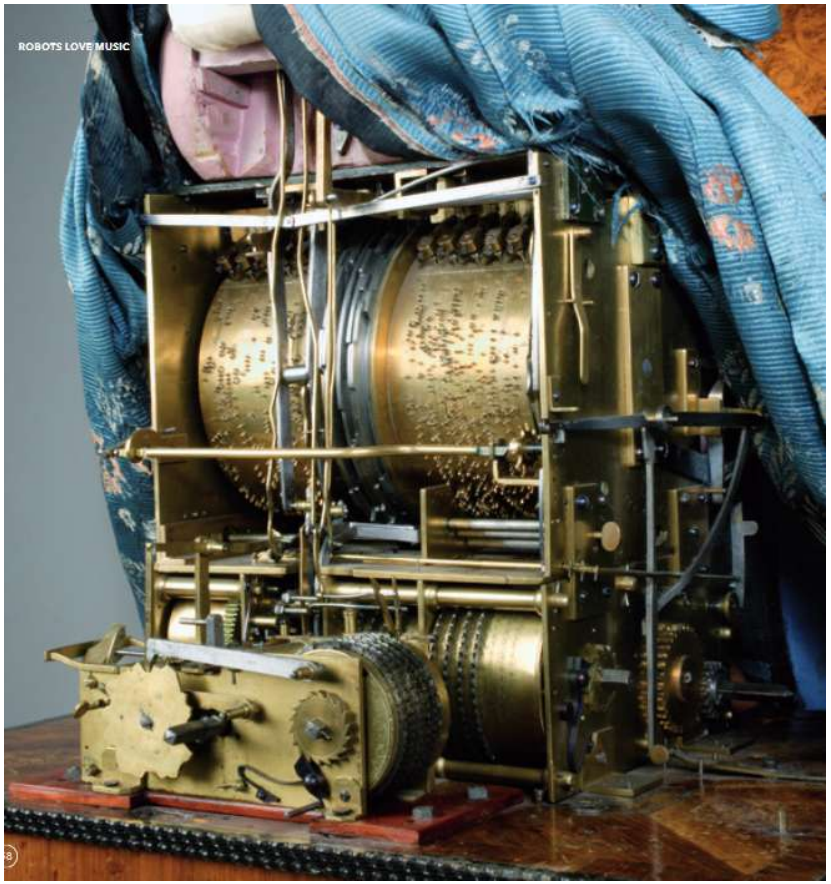
1981 – IBM prezentuje pierwszy komputer z systemem **MS DOS 1.0.**

1982 – SINCLAIR wypuszcza ZX SPECTRUM.

Automaty muzyczne

https://www.museumspeelklok.nl/

Automaty muzyczne



~1774

https://www.museumspeelklok.nl/

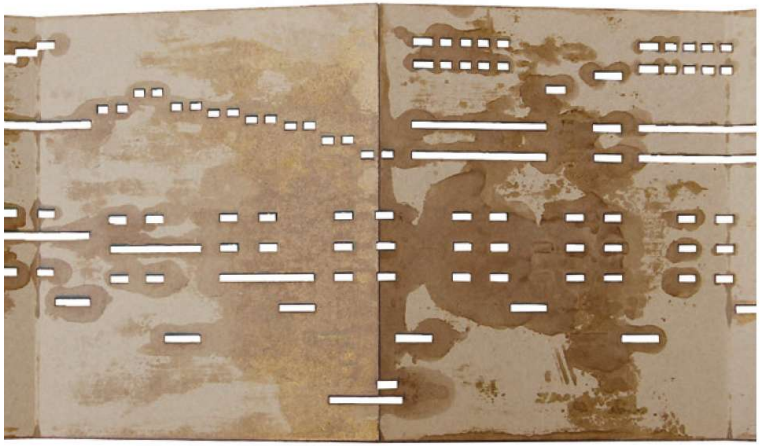
Automaty muzyczne

1838

1914

https://www.museumspeelklok.nl/

Automaty muzyczne

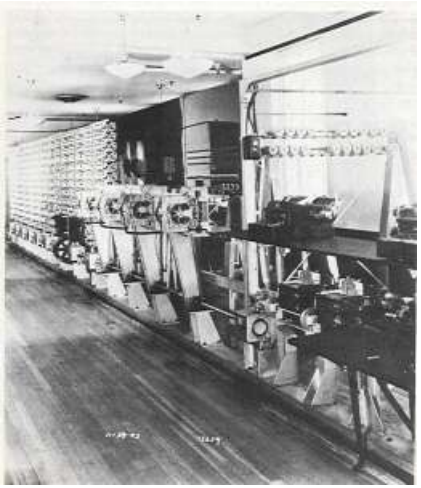


https://www.museumspeelklok.nl/

Generacje komputerów

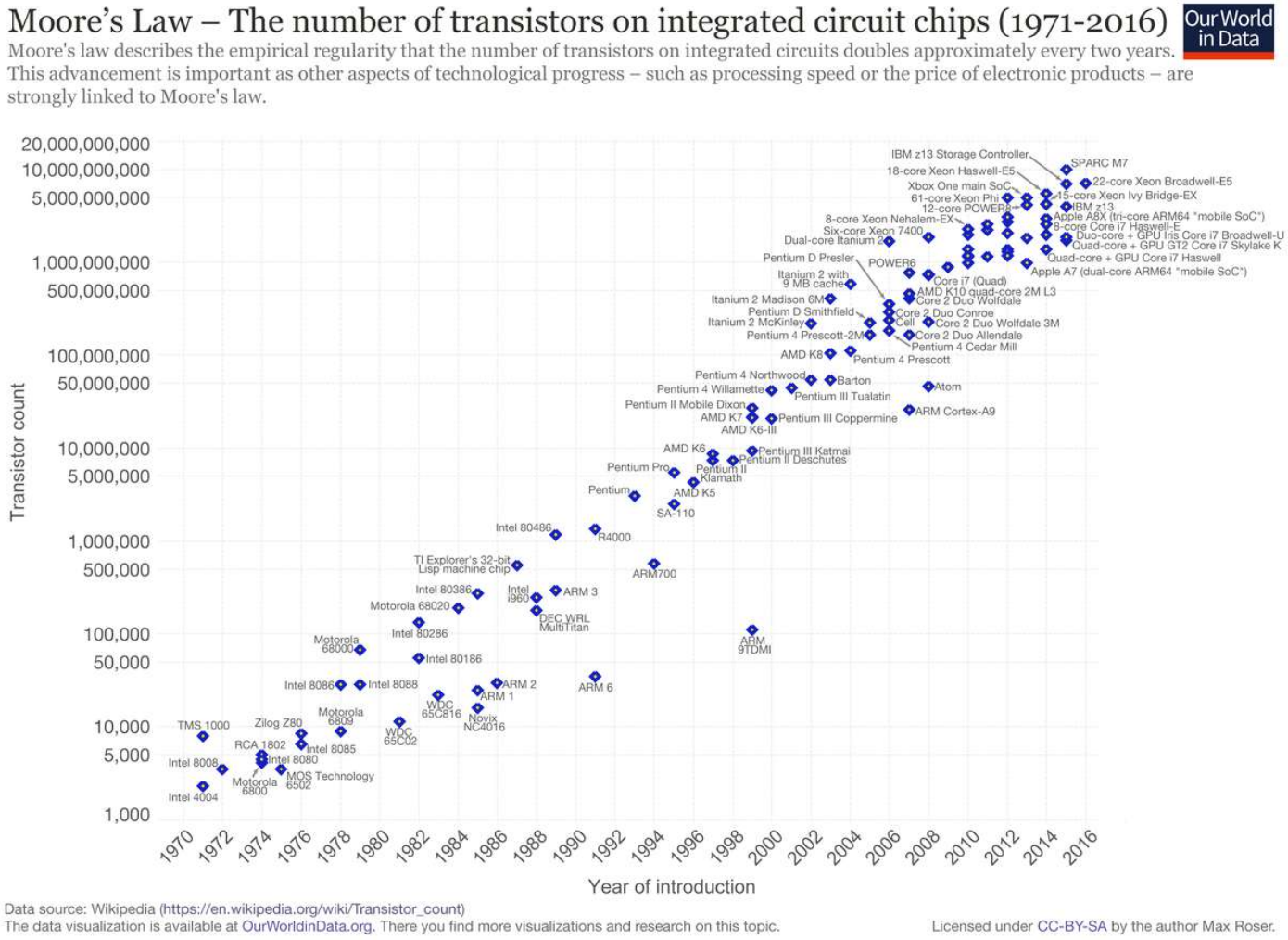
Generacje komputerów:

0 generacja – maszyny mechaniczne
I generacja – komputery lampowe
II generacja – komputery tranzystorowe
III generacja – komputery na układach scalonych
IV generacja – komputery w technologii VLSI (Very Large Scale Integration)

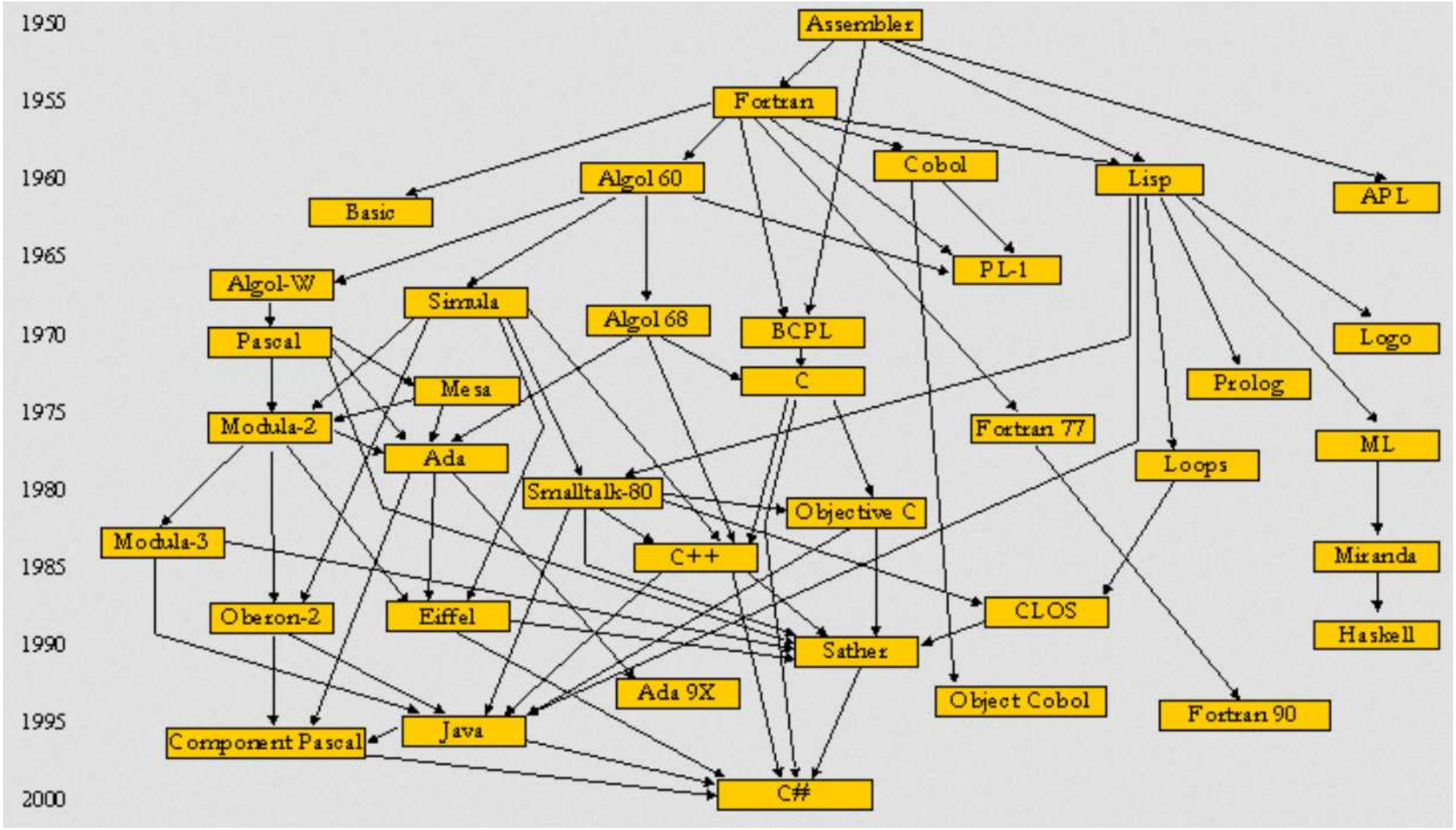


Prawo Moora

Prawo Moora – prawo stanowiące, że liczba tranzystorów w układach scalonych podwaja się co 18 miesięcy.



Ewolucja języków programowania



przykład drzewka obrazującego ewolucję pierwszych języków programowania

Generacje języków programowania

Pierwsza generacja – Programowanie pierwszych komputerów akceptujących zmianę oprogramowania odbywało się bezpośrednio w kodzie binarnym, który można przedstawić jako ciągi zer i jedynek. Każdy typ komputera operował własnym kodem (językiem maszynowym). Jest to główna wada tych języków, gdyż programista każdorazowo musiał dostosowywać się do języka konkretnej maszyny.

Przykład:

00101010010110110100010101101101011010001011100001000010011110000101010111110000101010100000101010000101010000010100101010010110110100010101101101011010001011100001000010011110000101011100001010101000010101000010101000001010001001011011010001010110110101101000101110000100001001111000010101110000101010100000101010000101010000010100010010

Generacje języków programowania

Druga generacja – Ponieważ operowanie ciągami zer i jedynek nie było wygodne dla programisty, przypisano im łatwiejsze do zrozumienia znaki mnemotechniczne. Tak narodziły się języki symboliczne, zwane też asemblerami. Choć stanowią proste tłumaczenie języka maszynowego na symbole i są ściśle związane z danym modelem komputera, to ułatwiają pisanie instrukcji i czynią je bardziej czytelnymi.

Przykład:

```
mov ax, 0D625h
mov es, ax
mov al, 24
mov ah, 0
int 21h
```

Generacje języków programowania

Trzecia generacja – Kolejnym krokiem w rozwoju języków programowania było powstanie języków wysokiego poziomu. Symbole asemblera reprezentujące konkretne instrukcje zostały zastąpione kodem nie związanym z maszyną, bardziej zbliżonym do języka naturalnego lub matematycznego.

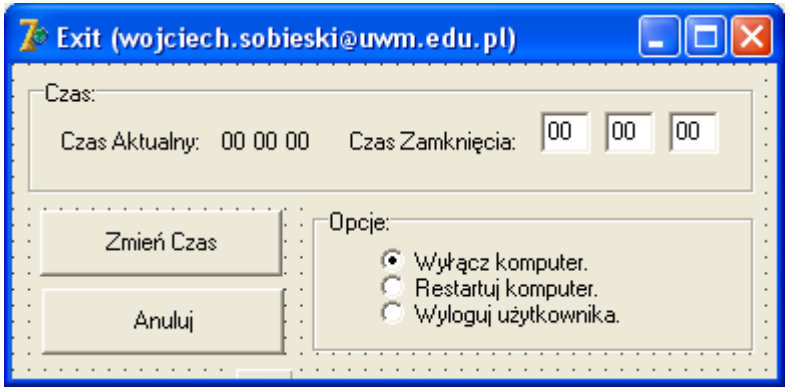
Przykład:

```
for i:=1 to MaxN do
   begin
   Vx[i]:=Vx[i]+dx[i];
   end;
```

Generacje języków programowania

Czwarta generacja – Na czwartą generację języków programowania składa się szereg narzędzi, które umożliwiają budowę prostych aplikacji przez zestawianie „prefabrykowanych" modułów. Obecnie wielu specjalistów uważa, że nie są to języki programowania w ścisłym znaczeniu, gdyż częstokroć stanowią jedynie rozszerzenie języków już istniejących. Niektórzy autorzy proponują stosować nazwę „czwarta generacja" wyłącznie w odniesieniu do programowania obiektowego.

Przykład:



Generacje języków programowania

Piąta generacja – Nazwę „język piątej generacji” stosuje się czasem w odniesieniu do języków używanych do tworzenia programów wykorzystujących tzw. sztuczną inteligencję (AI) lub inaczej – systemów ekspertowych.

Rankingi popularności

# Ranking	Programming Language	Percentage (YoY Change)	YoY Trend
1	JavaScript	19.014% (+0.225%)	
2	Python	16.351% (+0.243%)	
3	Java	12.817% (+2.086%)	
4	Go	7.574% (-1.348%)	
5	Ruby	7.194% (+0.702%)	
6	TypeScript	7.183% (-0.151%)	
7	C++	6.695% (-0.941%)	
8	PHP	4.910% (-0.288%)	
9	C#	3.413% (-0.383%)	
10	C	3.029% (-0.292%)	
39	Fortran	0.045%	

https://madnight.github.io/githut/#/

Fortran jest na pozycji 39
(5 października 2021)

Rankingi popularności

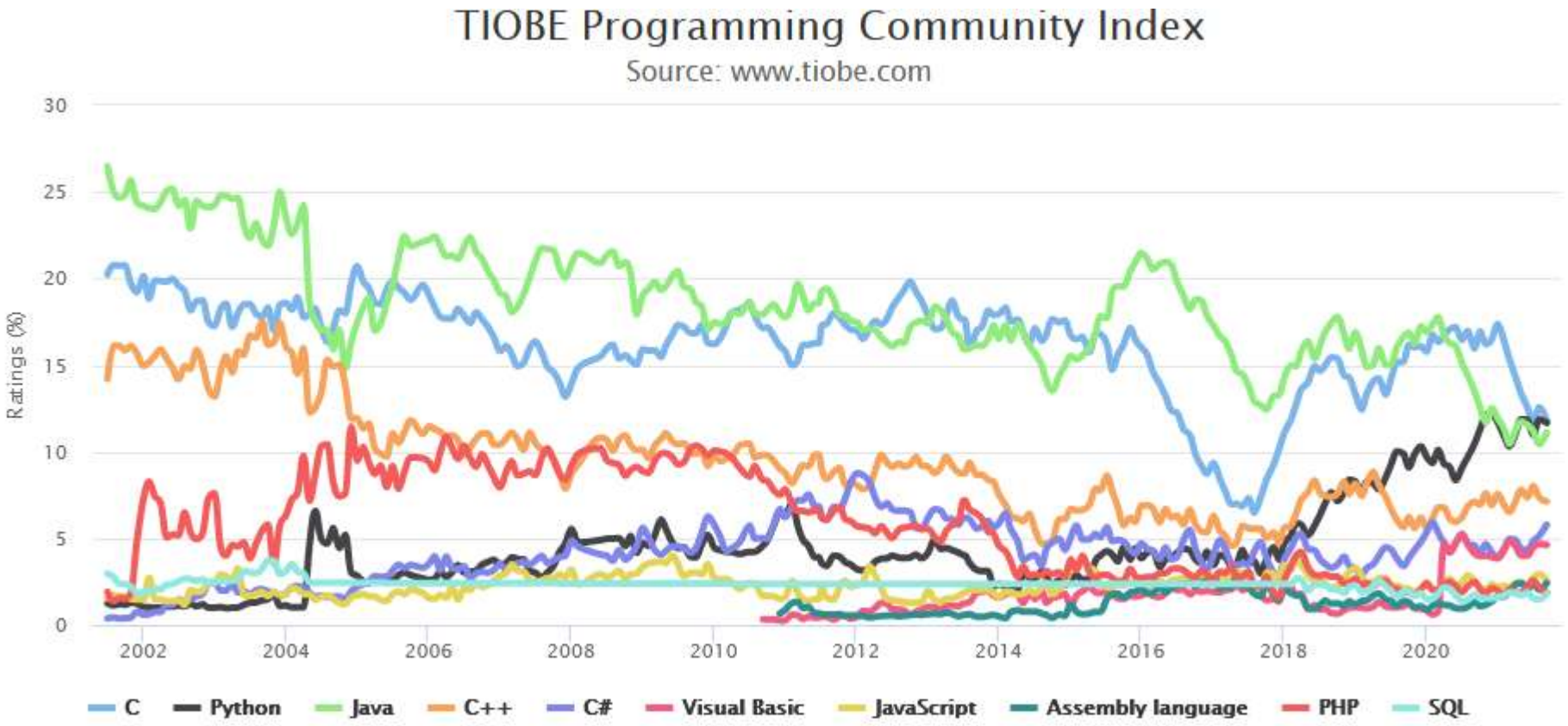
Sep 2021	Sep 2020	Change	Programming Language	Ratings	Change
1	1		C	11.83%	-4.12%
2	3	↑	Python	11.67%	+1.20%
3	2	↓	Java	11.12%	-2.37%
4	4		C++	7.13%	+0.01%
5	5		C#	5.78%	+1.20%
6	6		Visual Basic	4.62%	+0.50%
7	7		JavaScript	2.55%	+0.01%
8	14	⇈	Assembly language	2.42%	+1.12%
9	8	↓	PHP	1.85%	-0.64%
10	10		SQL	1.80%	+0.04%
11	22	⇈	Classic Visual Basic	1.52%	+0.77%
12	17	⇈	Groovy	1.46%	+0.48%
13	15	↑	Ruby	1.27%	+0.03%
14	11	↓	Go	1.13%	-0.33%
15	12	↓	Swift	1.07%	-0.31%
16	16		MATLAB	1.02%	-0.07%
17	37	⇈	Fortran	1.01%	+0.65%

Fortran jest na pozycji 17
(5 października 2021)

w roku 2019 był na pozycji 29

https://www.tiobe.com/tiobe-index/

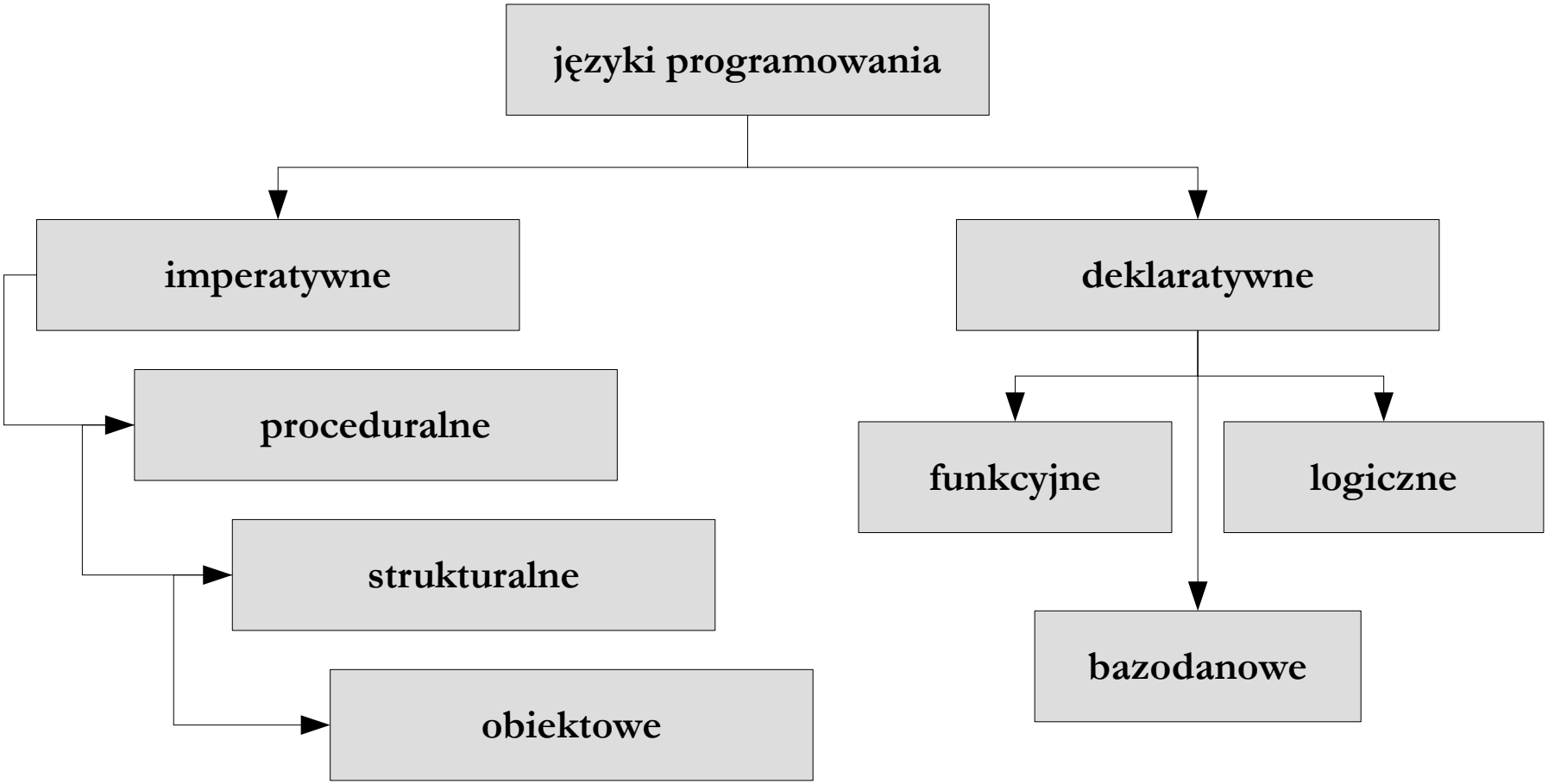
Rankingi popularności



https://www.tiobe.com/tiobe-index/

Paradygmaty programowania

Paradygmat programowania — sposób patrzenia programisty na przepływ sterowania i wykonywanie programu komputerowego.



Paradygmaty programowania

Języki imperatywne wymagają podania szczegółowej sekwencji czynności (rozkazów), jakie mają być wykonane do osiągnięcia celu.

Języki imperatywne są konsekwencją stosowanej obecnie architektury komputerów (von Neumanna).

Języki deklaratywne wymagają podania celu oraz zbioru zależności (nie podaje się jak cel ma być osiągnięty).

Języki deklaratywne stanowią próbę odtworzenia naturalnego sposobu myślenia człowieka.

Paradygmaty programowania

parzenie kawy

imperatywnie

- nalej do czajnika 0.8 litra zimnej wody z kranu w kuchni
- sprawdź czy czajnik jest podłączony
- włącz czajnik
- wyjmij 4 niebieskie filiżanki z drugiej półki szafki nad zlewem
- wyjmij puszkę kawy mielonej z trzeciej półki szafki koło lodówki
- ...

deklaratywnie

- zrób proszę 4 małe kawy

Paradygmaty programowania

Programowanie imperatywne – paradygmat programowania, który opisuje proces wykonywania jako sekwencję instrukcji zmieniających stan programu. Programy imperatywne składają się z ciągu komend do wykonania przez komputer.

```
...
dx = lx/(nx-1)
do i =1, nx
  x(i) = (i-1)*dx
end do
...
```

Paradygmaty programowania

Programowanie funkcyjne – paradygmat programowania zalecający dzielenie kodu na funkcje, czyli fragmenty wykonujące ściśle określone operacje. W tym ujęciu funkcją jest nawet program główny, którego zakończenie powoduje zwrot wartości: zero oznacza, że program wykonał się bez żadnych błędów.

```
#include <stdio.h>
int main()
{
  printf("Hello World");
  return 0;
}
```

Paradygmaty programowania

Programowanie proceduralne – paradygmat programowania zalecający dzielenie kodu na procedury, czyli fragmenty wykonujące ściśle określone operacje. Procedury nie powinny korzystać ze zmiennych globalnych (w miarę możliwości), lecz pobierać i przekazywać wszystkie dane (czy też wskaźniki do nich) jako parametry wywołania.

…
call czytaj_parametry_zadania(…)
call generuj_siatke(…)
call definiuj_warunki-brzegowe(…)
call definiuj_warunki_poczatkowe(…)
...

Paradygmaty programowania

Programowanie strukturalne – paradygmat programowania zalecający hierarchiczne dzielenie kodu na moduły, które komunikują się jedynie poprzez dobrze określone interfejsy. Jest to rozszerzenie koncepcji programowania proceduralnego.

Programowanie strukturalne wykorzystuje trzy rodzaje konstrukcji:

- **sekwencja** – ciąg instrukcji niezależny od stanu programu
- **wybór** – ciąg instrukcji zależny od stanu programu
- **iteracja** – powtarzanie ciągu instrukcji aż do spełnienia jakiegoś warunku

Programowanie strukturalne nie dopuszcza stosowanie instrukcji **skoku**.

Paradygmaty programowania

Programowanie obiektowe – paradygmat programowania, w którym programy definiuje się za pomocą obiektów: elementów łączących stan (czyli dane) i zachowanie (czyli procedury, tu: metody). Obiektowy program komputerowy wyrażony jest jako zbiór takich obiektów, komunikujących się pomiędzy sobą w celu wykonywania zadań.

Podejście to różni się od tradycyjnego programowania proceduralnego, gdzie dane i procedury nie są ze sobą bezpośrednio związane.

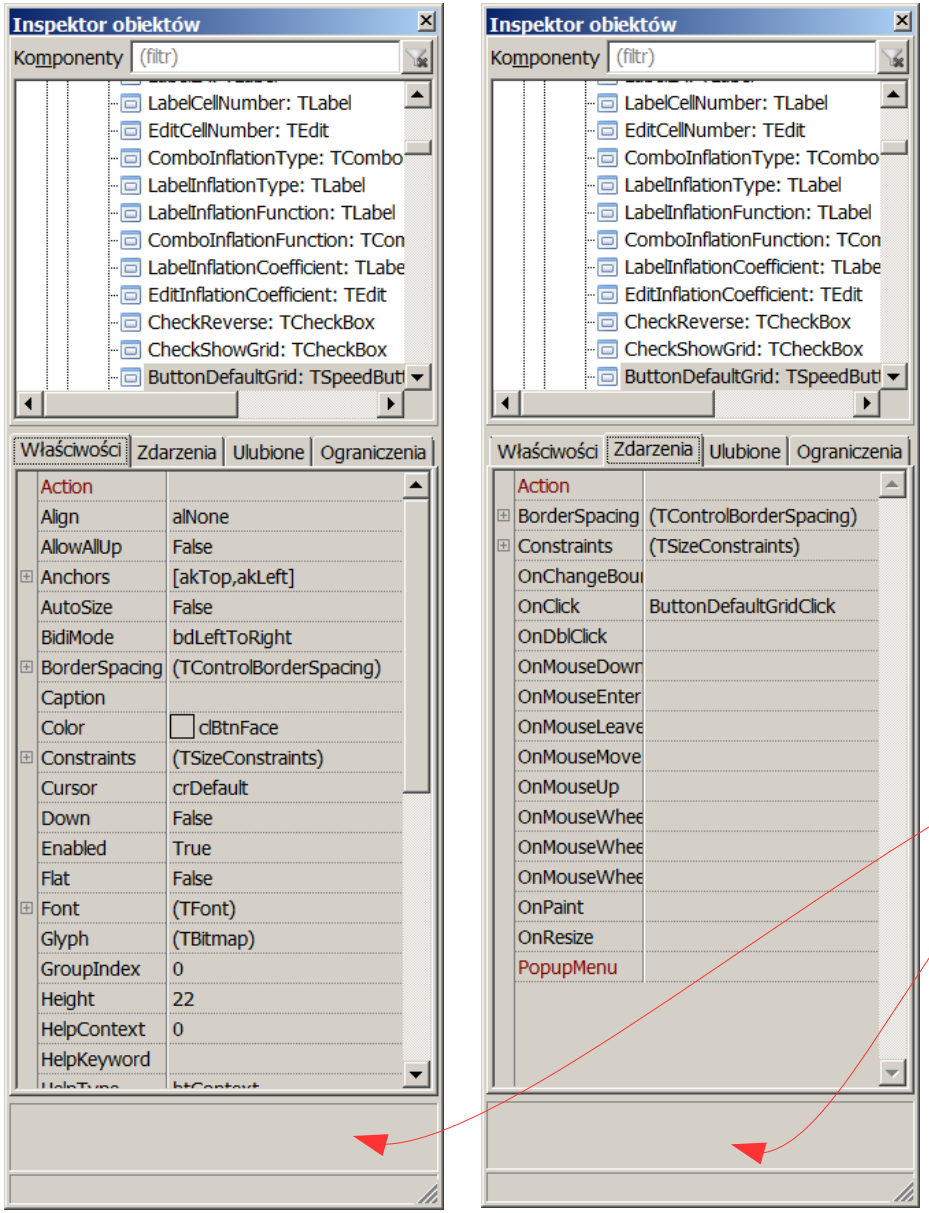
Programowanie obiektowe sprawdza się szczególnie dobrze do tworzenia **Graficznych Interfejsów Użytkownika**.

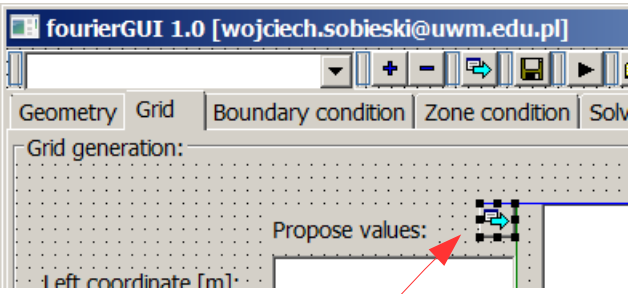
Paradygmaty programowania

Programowanie wizualne – paradygmat programowania (wariant programowania obiektowego), w którym programy definiuje się za pomocą **wizualnych komponentów** (obiektów), którym przypisane są określone **właściwości** oraz **zdarzenia**.

Programowanie wizualne nosi również nazwę **Rapid Application Development** (RAD), ze względu na możliwość szybkiego tworzenia graficznego interfejsu użytkownika. Wadą podejścia RAD jest trudność w zarządzaniu dużymi projektami.

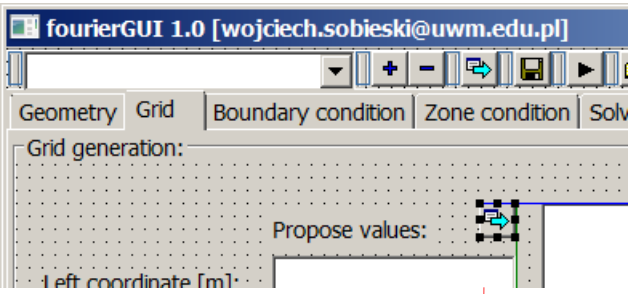
Paradygmaty programowania

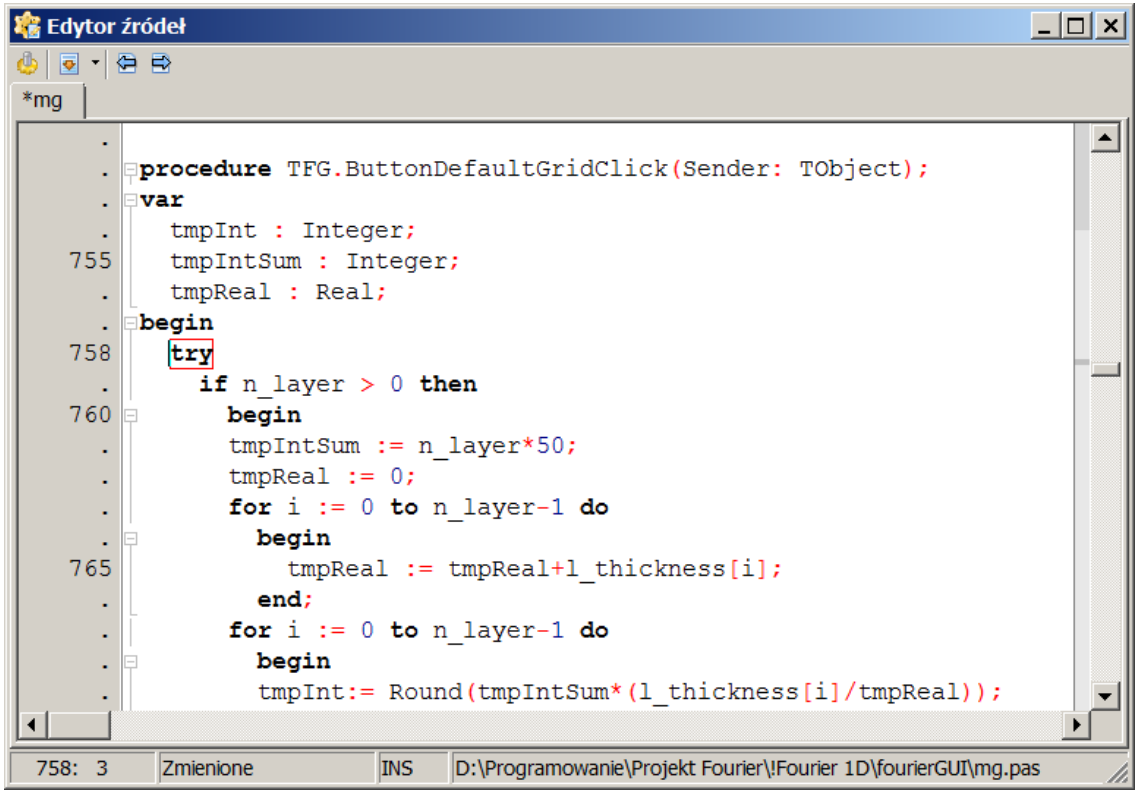




obiekt (tu: klawisz) posiada **zbiór właściwości** oraz **zbiór metod**, czyli „rzeczy”, które można z nim zrobić (fragment programu fourierGUI)

Paradygmaty programowania





procedura obsługi **zdarzenia**:
pojedyncze kliknięcie na klawisz

UNIVERSITY OF WARMIA AND MAZURY IN OLSZTYN
The Faculty of Technical Sciences
POLAND, 10-957 Olsztyn, M. Oczapowskiego 11
tel.: (48)(89) 5-23-32-40, fax: (48)(89) 5-23-32-55
URL: http://www.uwm.edu.pl/edu/sobieski/ (in Polish)

Dziękuję

Wojciech Sobieski

Olsztyn, 2001-2021