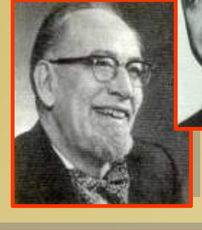
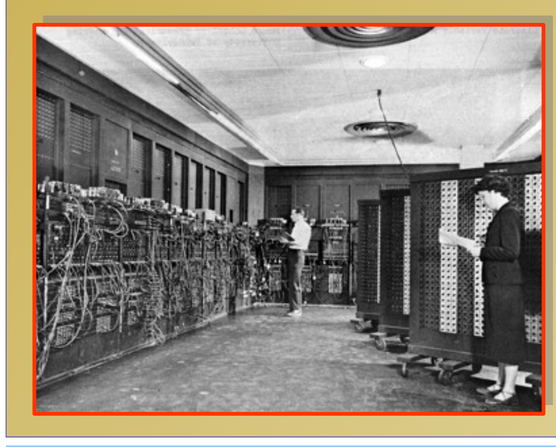
Покоління комп’ютерів

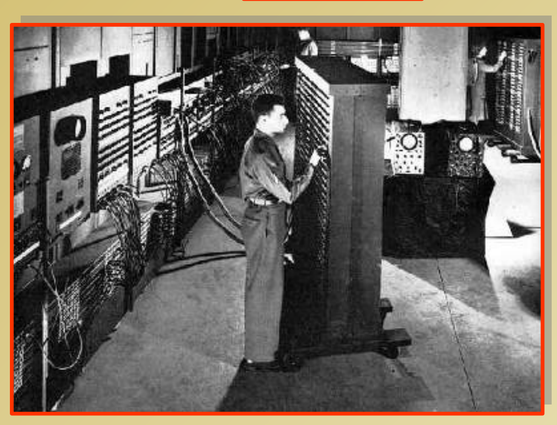
1945 – 1955 Електронно-вакуумні лампи

швидкодія **10-20 тис.** операцій в секунду
кожна машина має свою мову,
немає операційних систем
ввід и вивід: перфоленти, перфокарти, магнітні

ЕНІАК (1946)*Electronic Numerical Integrator And Computer*
Дж. Моучлі і П. Еккерт
Перший комп’ютер загального призначення на електронних лампах:
довжина 26 м, вага 35 тонн
додавання – 1/5000 сек, ділення – 1/300 сек
Десяткова система зчислення
10-разрядні числа



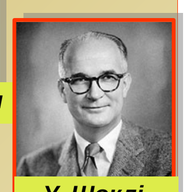




1955 – 1965 Транзистори

на напівпровідникових **транзисторах**
(1948, *Дж. Бардин*, *У. Бреттейн* и *У. Шоклі*)
10-200 тис. операцій в секунду
перші **операційні системи**
перші **мови програмування**: *Фортран* (1957), *Алгол* (1959)
Засоби збереження інформації:
магнітні барабани, **магнітні диски**

Дж. Бардин



У. Шоклі

1953-1955. IBM 604, IBM 608, IBM 702
1965-1966. БЭСМ-6
60 000 транзисторів
200 000 діодів
1 млн. операцій
в секунду
память – магнітна лента, магнітний барабан
працювали до 90-х р.р.

1965 – 1980 інтегральні мікросхеми

на і**нтегральних мікросхемах** (1958, *Дж. Кілбі*)
швидкодія до **1 млн.** операцій за секунду
оперативна пам’ять – **сотни Кбайт**
операційні системи – керування пам’яттю, пристроями, часом процесора
мови програмування ***Бейсік*** (1965),
Паскаль (1970, *Н. Вірт*), ***Си*** (1972, *Д. Рітчі*)
сумісність програм

Великі універсальні комп’ютери

1964. IBM/360 фірми IBM.
кеш-пам’ять
конвейєрна обрабітка команд
операційна система OS/360
1 байт = 8 біт (а не 4 або 6!)
Розподіл часу
1970. IBM/370
1990. IBM/390
1971. ЕС-1020 20 тис. оп/с Пам'ять 256 Кб
1977. ЕС-1060 1 млн. оп/с Пам’ять 8 Мб
1984. ЕС-1066 5,5 млн. оп/с Пам'ять 16 Мб

с 1980 по … Великі інтегральні схеми

компьютери на великих і зверхвеликих інтегральних схемах (**БІС**, **СБІС**)
суперкомпьютери
персональні компьютери
больше **1 млрд**. операцій в секунду
оперативна память – до декількох **гігабайт**
багатопроцесорні системи
компьютерні **мережі**
мультимедіа (графіка, анімація, звук)

Суперкомп’ютери

1972. ILLIAC-IV (США)
20 млн. оп/с, багатопроцесорна система
1976. Cray-1 (США)
166 млн. оп/с, память 8 Мб
1980. Ельбрус-1 (СРСР)
15 млн. оп/с, память 64 Мб
1985. Ельбрус-2
8 процесорів, 125 млн. оп/с, память 144 Мб
1985. Cray-2 2 млрд. оп/с
1989. Cray-3 5 млрд. оп/с
1995. GRAPE-4 (Японія)
1692 процесора, 1,08 трлн. оп/с
2002. Earth Simulator (NEC)
5120 процесорів, 36 трлн. оп/с
2007. BlueGene/L (IBM)
212 992 процесора, 596 трлн. оп/с

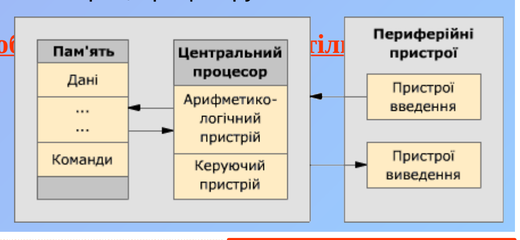
Принципи фон Неймана

Джон фон Нейман

Принцип двійкового кодування: вся інформація кодується в двійковому виді.
Принцип програмного керування:
програма складається з набору команд, які виконуються процессором автоматично кроком в певній послідовності.
Принцип однородності памяті:
програми і дані зберігаються в одній і тій же памяті.
Принцип адресності: память складається з пронумерованих комірок, процессору в любий момент часу доступна люба комірка.

Революційність ідеї фон Неймана полягала у тому, щоб … і команди для їх обробки.

Схема будови комп'ютера

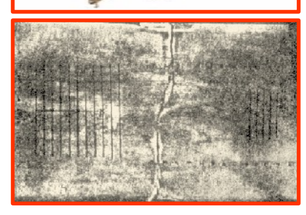


Давні засоби рахунків

Кости с зарубками («вестоницкая кость», Чехия, 30 тыс. лет до н.э)

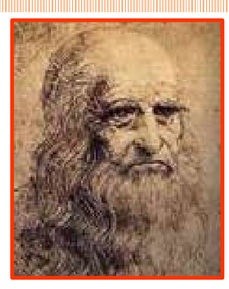
Узелковое письмо (Южная Америка, VII век н.э.)
узлы с вплетенными камнями
нити разного цвета (красная – число воинов, желтая – золото)
десятичная система

Саламинская доска (о. Саламин в Эгейском море (300 лет до н.э.))
бороздки – единицы, десятки, сотни, …
количество камней – цифры
десятичная система

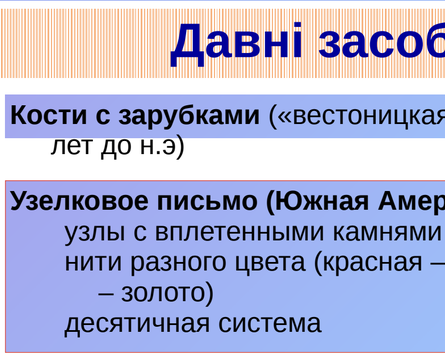


Перші проекти лічильних машин

Леонардо да Вінчі (XV ст.) – **сумуючий пристрійз зубчатими колесами: додавання 13-розрядних чисел**

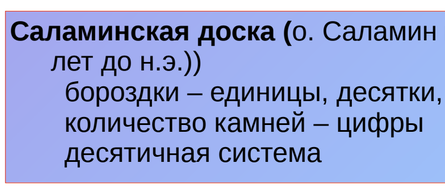


Вільгельм Шиккард (XVI ст.) – **сумуючий «лічильний годинник»: додавання і множення 6-розрядних чисел (машина була побудована, але згоріла)**



«Паскалина» (1642)

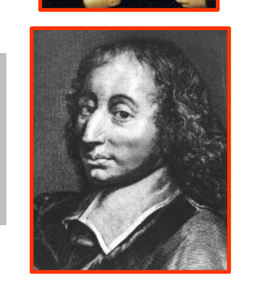
Блез Паскаль (1623 - 1662) **машина побудована, зубчасті колеса, додавання і віднімання 8-розрядних чисел, десяткова система**

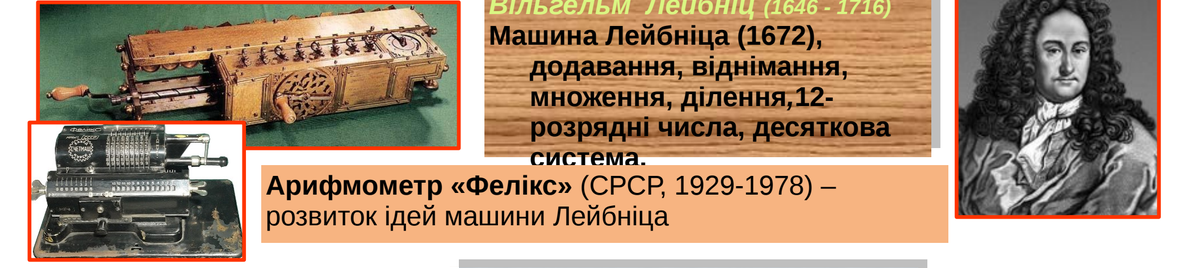


Вільгельм Лейбніц (1646 - 1716)
Машина Лейбніца (1672), додавання, віднімання, множення, ділення,12-розрядні числа, десяткова система.

Арифмометр «Фелікс» (СРСР, 1929-1978) – розвиток ідей машини Лейбніца

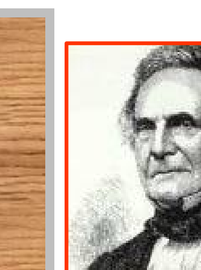
Чарльз Бебідж (1792-1871)
Різницева машина (1822)
Аналітична машина (1834)
«мельниця» (автоматичне виконання обчислень)
«склад» (зберігання даних)
«контора» (керування)
ввід даних і програми з перфокарт
ввід програми «на ходу»



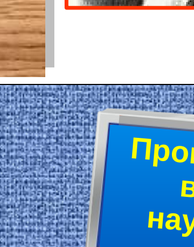


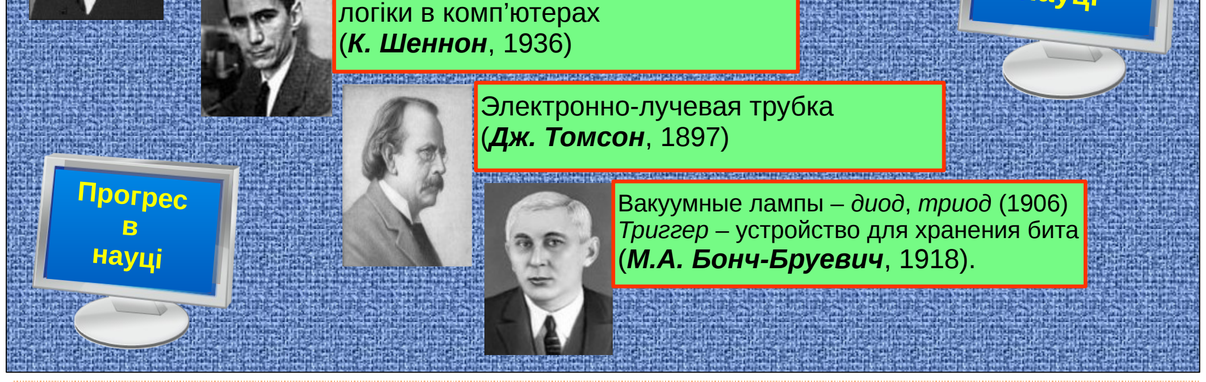
Перші компютери

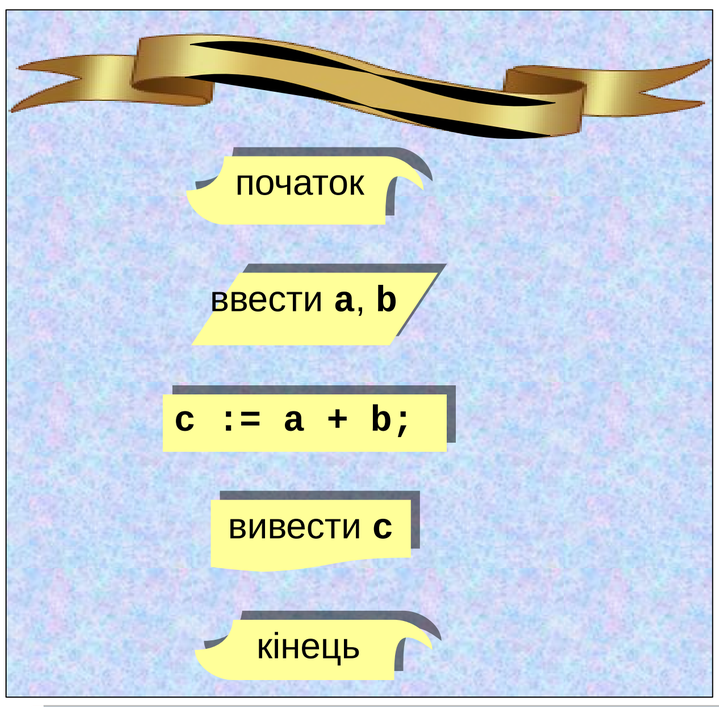
1937-1941. *Конрад Цузе*: *Z1, Z2, Z3, Z4.*
Електромеханічне реле (пристрій з двома станами), двійкова система, використання бульової алгебри, ввід даниxx з кіноленти.
1939-1942. Первый макет электронного лампового компьютера, *Дж. Атанасофф*
двійкова система, розвязування систем 29 лінійнних рівнянь
Марк-I (1944)
Разработчик – *Говард Айкен* (1900-1973)
Первый компьютер в США:
длина 17 м, вес 5 тонн, 75 000 электронных ламп, 3000 механических реле, сложение – 3 секунды, деление – 12 секунд



Джон Атанасофф







Структура програми на Паскалі

```
program <ім'я програми>;
const …;{константи}
var …; {змінні}
 { процедури і функції }

begin
… {основна програма}
end.
```

коментарі у фігурних дужках
не опрацьовуються

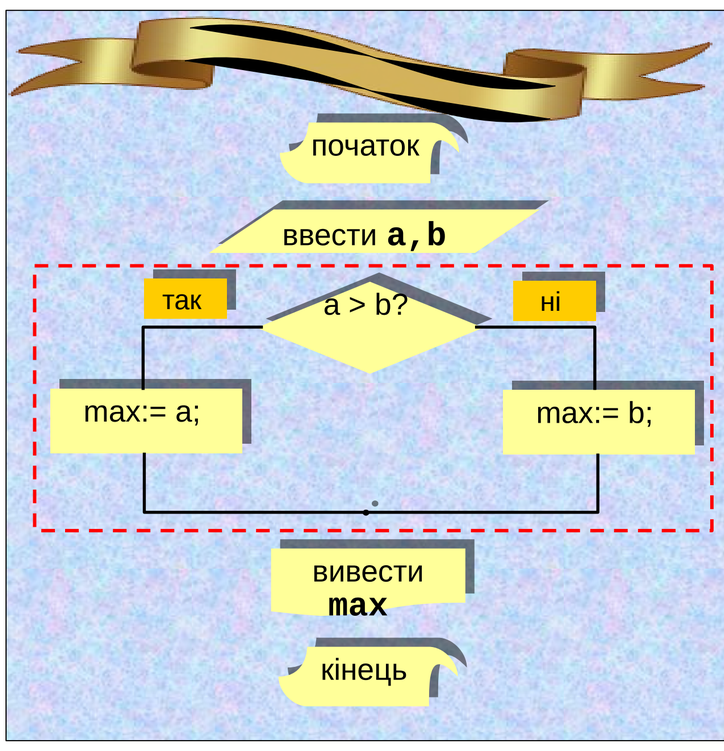
```
   program qq;
   var a, b, c: integer;
   begin
writeln('Ввести два цілих числа');
read ( a, b );
c := a + b;
writeln ( a, '+', b, '=', c );
   end.
```

Протокол: `Ввести два цілих числа`

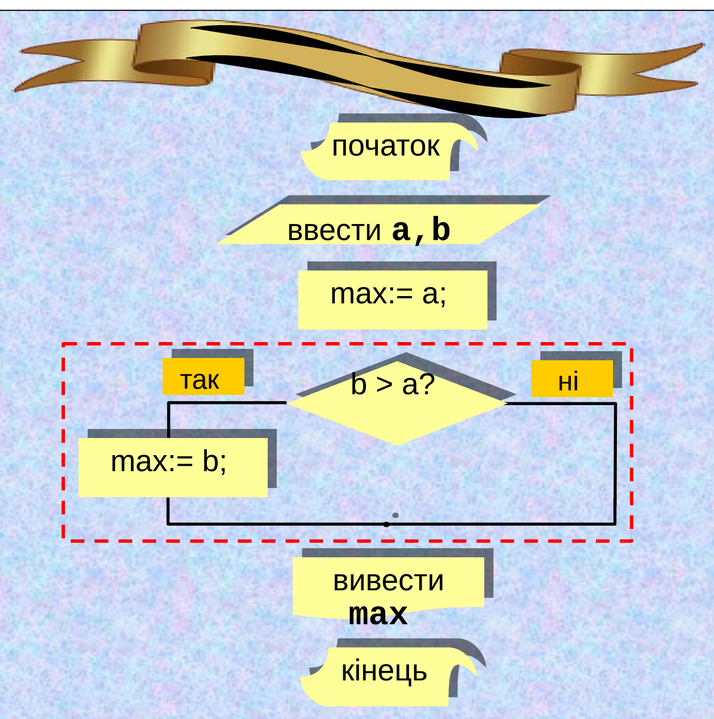
`25 30` — це вводить користувач

`25+30=55` — це виводить комп'ютер

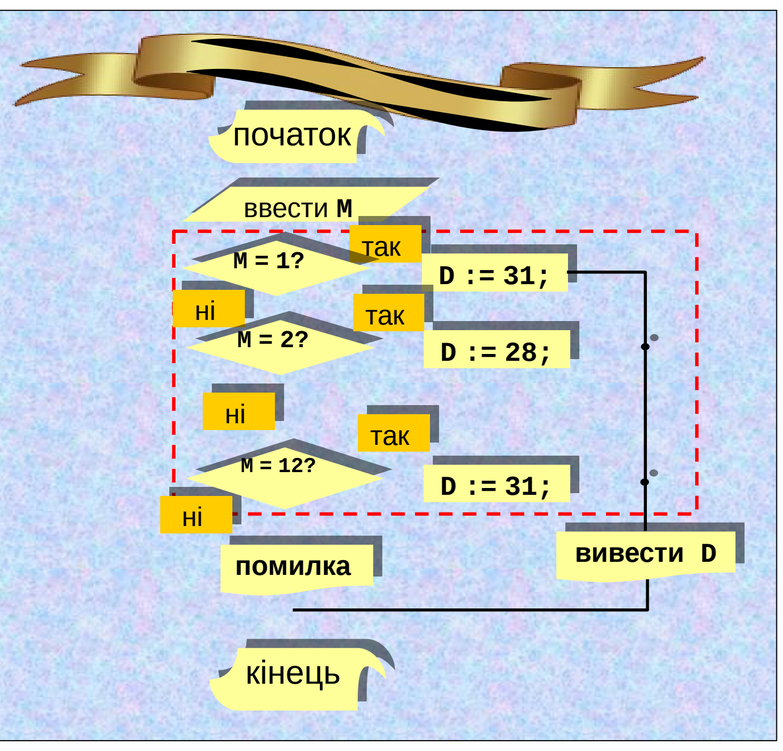
Р О З Г А Л У Ж Е Н Н Я




```
   program qq;
   var a, b, max: integer;
   begin
writeln('Ввести два цілих числа');
read ( a, b );
max := a;
if b > a then
   max := b;
writeln ('Більше число ', max);
   end.
```

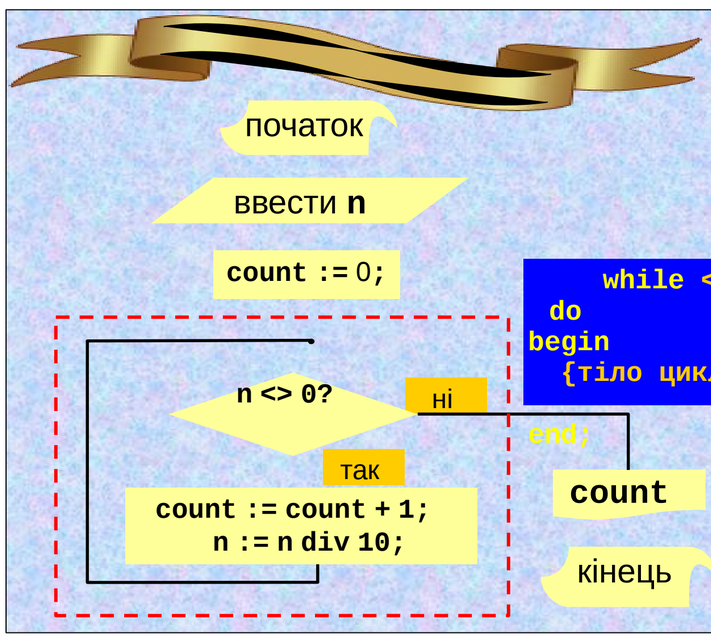



```
   program qq;
   var a, b, max: integer;
   begin
writeln('Ввести два цілих числа');
read ( a, b );
if a > b then begin
   max := a;
end
else begin
   max := b;
end;
writeln ('Більше число ', max);
   end.
```

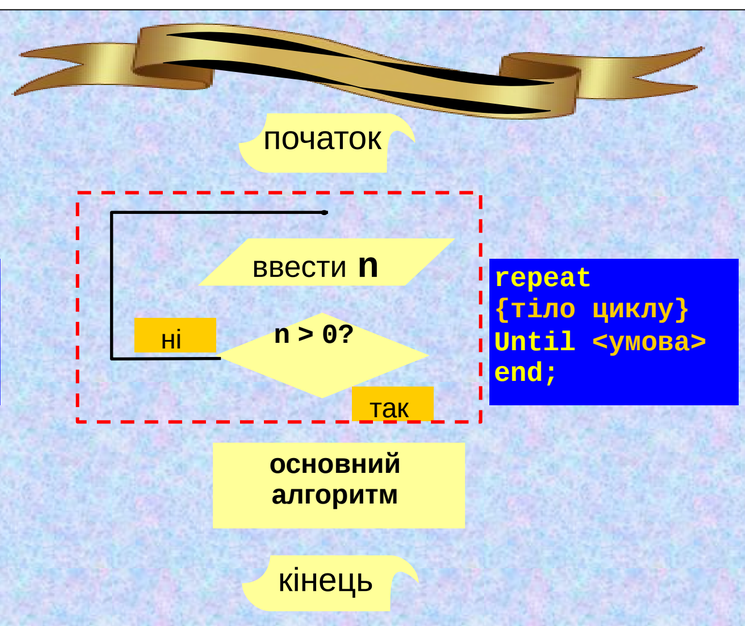



```
program qq;
var M, D: integer;
begin
 writeln('Ввести номер місяця:');
 read ( M );
case M of
  2:         begin D := 28; end;
  4,6,9,11: begin D := 30; end;
  1,3,5,7,8,10,12: D := 31;
  else              D := -1;
end;
 if D > 0 then
    writeln('В цьому місяці ', D, ' днів.')
 else
    writeln('Неправильний номер місяця');
end.
```

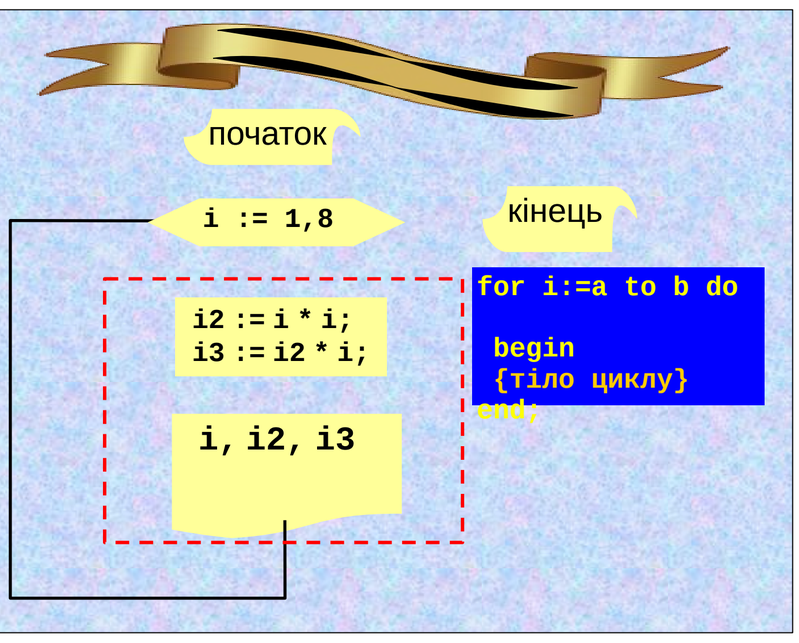
Ц И К Л И




```
    while <умова>
 do
begin
  {тіло циклу}
end;
```




```
repeat
{тіло циклу}
Until <умова>
end;
```




```
for i:=a to b do

 begin
 {тіло циклу}
end;
```

```
program qq;
var n, count , n1: integer;
begin
 writeln('Ввести ціле число');
 read(n);
 n1 := n;
 count := 0;
 while n <> 0 do begin
   count := count + 1;
   n := n div 10;
 end;
 writeln('В числі ', n1, ' знайшли ',
          count, ' цифр');
end.
```

```
program qq;
var n, count: i , n1: integer;
begin
 writeln('Ввести ціле число');
 read(n);
 count := 0;
 n1 := n;
 repeat
  count := count + 1;
  n := n div 10;
 until n>0;
 writeln('В числі ', n, n1, ' ішли ',
          count, ' цифр');
end.
```

```
program qq;
var i, i2, i3: integer;
begin
 for i:=1 to 8 do begin
   i2 := i*i;
   i3 := i2*i;
   writeln(i:4, i2:4, i3:4);
 end;
end.
```

Компютерний вірус - спеціально написана програма, яка може самовільно приєднуватися до других програм, створювати свої копії і записувати їх у файли, системні області компютера і в обчислювальні мережі с ціллю порушення роботи програм, псування файлів та каталогів, створення перешкод у нормальній роботі компютера.

К л а с и ф і к а ц і я в і р у с і в

- **Середовище перебування**
  - Мережеві
  - Файлові
  - Заватажувальні
  - Файлово-завантажувальні
  - Макро
- **Спосіб зараження середовища перебування**
  - Резидентні
  - Нерезидентні
- **Степінь дії**
  - Нешкідливі
  - Безпечні
  - Небезпечні
  - Дуже небезпечні
- **Особливості алгоритму**
  - Прості
  - Реплікатори (черви)
  - Стелс (невидимки)
  - Поліморфні (мутанти)
  - Квазивірусні (троянскі)

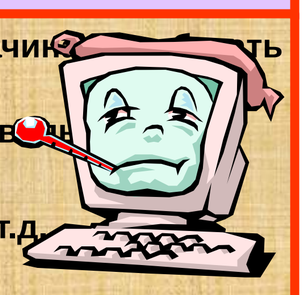
Середовище перебування віруса	
Мережеві	Поширюються по різним комютерним мережам.
Файлові	Впроваджуються в виконуючі модулі, тобто в файли, які мають расширення COM або EXE. Можуть впроваджуватися і в другі типи файлів, але записані в таких файлах, вони ніколи не одержують керування і гублять властивість до розможення.
Завантажувальні	Впроваджуються в загрузочний сектор диска (Boot-сектор) або в сектор, який містить програму загрузки системного диска (Master Boot Record).
Файлово-завантажувальні	Заражають як файли, так і загрузочні сектори дисків.
Макро	Заражають файли-документи і електронні таблиці деяких популярних редакторів.

Спосіб зараження оточуючого середовища	
Резидентні	При зараженні (інфікуванні) компьютера залишає в памяті комп'ютера свою резидентну частину, яка потім перехоплює звернення операційної системи до об'єктів зараження (файлам, загрузочним секторам дисків і т. д.) і впроваджується в них. Резидентні віруси знаходяться в памяті і залищаються активними до виключення або перезагрузки компьютера.
Нерези-дентні	Не заражають память компютера і є активними обмежений час.

Спосіб впливу віруса	
Нешкідливі	Не впливають на роботу компютера (крім зменшення вільної памяті на диску в результаті свого поширення).
Безпечні	Вплив обмежується зменшенням вільної памяті на диску і графічними, звуковими та іншими ефектами.
Небезпечні	Можуть привести до серйозних сбоїв в работі компютера.
Дуже небезпечні	Вплив вірусів може призвести до втрати програм, знищення даних, стиранню інформації в системних областях диска.

Особливості алгоритму	
Простей-шие	Изменяют содержимое файлов и секторов диска и могут быть легко обнаружены и уничтожены.
Репликаторы (черви)	Распространяются по компьютерным сетям, вычисляют адреса сетевых компьютеров и записывают по этим адресам свои копии.
Стелс (невидимки)	Очень трудно обнаружить и обезвредить, так как они перехватывают обращения операционной системы к пораженным файлам и секторам дисков и подставляют вместо своего тела незараженные участки диска
Полиморфные (мутанты)	Содержат алгоритмы шифровки-расшифровки, благодаря которым копии одного и того же вируса не имеют ни одной повторяющейся цепочки байтов.
Квазивирусные (троянские)	Не способны к самораспространению, но очень опасны, так как, маскируясь под полезную программу, разрушают загрузочный сектор и файловую систему дисков.

Ознаки зараження компютерним вірусом



некоторые программы перестают работать или начинают работать неправильно;

на экран выводятся посторонние сообщения, символы;

работа на компьютере существенно замедляется;

некоторые файлы оказываются испорченными и т.д.

операционная система не загружается;

изменение даты и времени модификации файлов;

изменение размеров файлов;

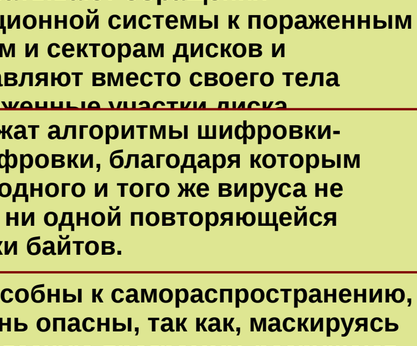
значительное увеличение количества файлов на диска;

существенное уменьшение размера свободной оперативной памяти и т.п.

Методи захисту

- Програмні
- Апаратні
- Організаційні

Захист від компютерних вірусів



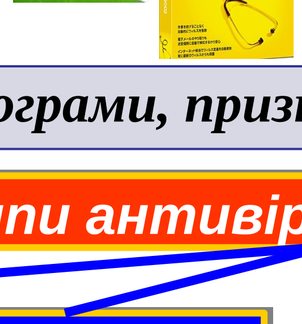
Перед використанням чужих дисків обов'язково перевіряйте їх на наявність вірусів. Не запускайте неперевірені файли, одержані електронною поштою.

Слід регулярно виконувати копіювання цінної інформації. При копіюванні на диски бажано мати дві різні версії, щоб підвищити надійність цих носіїв.

Необхідно мати завантажувальний диск із записом операційної системи та програмою. Диск має бути захищена від запису.

Виконуйте періодичну перевірку пам'яті та всіх дисків за допомогою нових версій антивірусних програм.

Вчасно оновлюйте свої антивірусні програми. Тільки при постійному оновленні версій антивірусних програм можна встигнути за «творцями» нових вірусів і бути впевненими, що ваші дані й диски не будуть уражені.



Антивіруси* — *це програми, призначені для виявлення і знешкодження вірусів.

Типи антивірусних програм

Програми-детектори перебування

Поділяються на детектори, що дозволяють виявляти і вилучати відомі віруси, і детектори, спроможні боротися із досі невідомими (тобто новими) вірусами. До першої групи детекторів належить популярна в минулі роки програма ***Aidstest***, розроблена Д. М. Лозинським. Детектори другої групи містять так званий евристичний аналізатор, спроможний виявляти віруси, про які ще не знали автори детектора на момент його розробки і які можуть з'явитися згодом. Прикладом евристичного детектора є потужна антивірусна програма ***DrWeb.***

Програми-ревізори

Ці програми контролюють усі вразливі (для вірусної атаки) компоненти комп'ютера. Принцип їхньої дії полягає у тому, що вони запам'ятовують дані про стан файлів і системних ділянок дисків, а при наступних запусках порівнюють їхній стан

із вихідним.

Програми-охоронці

Подібні програми резидентно розташовуються в пам'яті комп'ютера й автоматично перевіряють на наявність вірусів файли, які запускаються, і дискети, що вставляються до дисковода. При виявленні вірусу програма-охоронець може видавати попереджувальне повідомлення, а також може запобігти тим діям вірусу, які можуть призвести до його розмноження або зашкодити

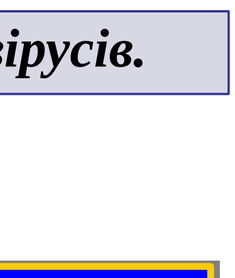
системі.

Антивірусні комплекси

. Сучасні антивірусні програми — це комплекси, що поєднують функції детектора, ревізора й охоронця. До таких комплексів належить широко відома програма *Norton Antivirus*, а також пакет *Anti-Viral Toolkit Pro* (скорочено *AVP*). Останній — найпопулярніший у країнах СНД — створений у лабораторії Є. Касперського. В Інтернеті можна знайти безплатний український антивірусний комплекс

Zillya.

ВИСНОВОК: **Як же вберегти комп'ютер від вірусу? Треба робити так само, як і з людськими хворобами: найкращий спосіб запобігти хворобі — не бути зараженим. Найважливіше правило — не користуватися флешами та компакт-дисками невідомого походження. Саме вони в більшості випадків стають розповсюджувачами вірусів. Щоразу перед тим, як прочитати дані з флешки або диска, необхідно перевіряти її на наявність вірусу спеціальною антивірусною програмою.**



Масив – це група однотипних елементів, які мають спільне ім'я і розміщені в пам'яті поряд.

Оголошення масиву

ім'я | початковий індекс | кінцевий індекс | тип елементів

```
var A: array[ 1 .. 5 ] of integer ;
```

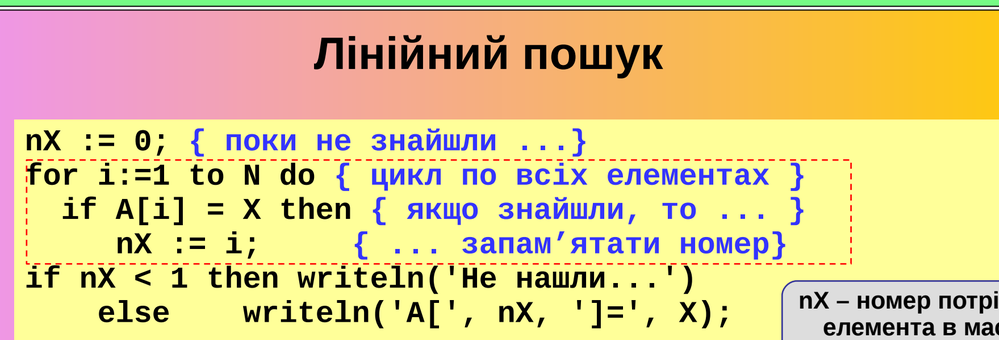
```
var X, Y: array [1..10] of real;
    C: array [1..20] of char;
```

```
var A: array ['A'..'Z'] of real;
    B: array [False..True] of integer;
```

Значення та індекс елементів масиву

ЗНАЧЕННЯ елемента масиву

НОМЕР елемента масиву (**ІНДЕКС**)



Пошук у масиві

Задача – знайти в масиві елемент, рівний X, або встановити, що його немає.

Лінійний пошук

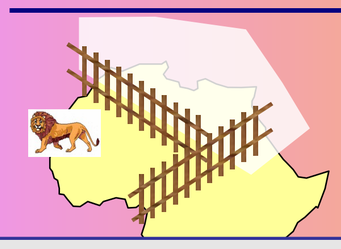
```
nX := 0; { поки не знайшли ...}
for i:=1 to N do { цикл по всіх елементах }
  if A[i] = X then { якщо знайшли, то ... }
     nX := i;     { ... запам'ятати номер}
if nX < 1 then writeln('Не нашли...')
    else    writeln('A[', nX, ']=', X);
```

nX – номер потрібного елемента в масиві

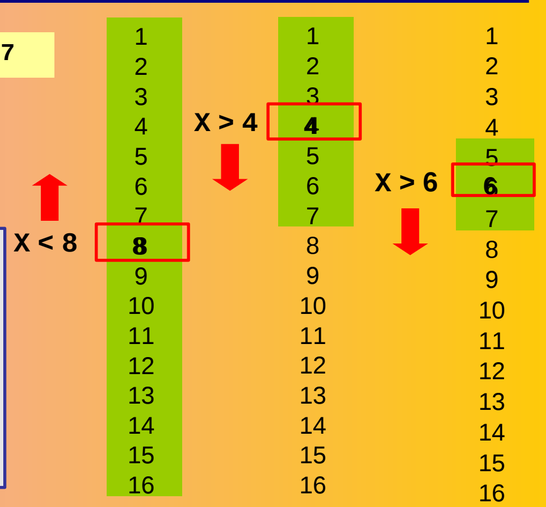
```
nX := 0;
for i:=1 to N do
if A[i] = X then begin
  nX := i;
  break; {вихід з циклу}
end;
```

```
nX := 0; i := 1;
while i <= N do begin
 if A[i] = X then begin
   nX := i; i := N;
 end;
 i := i + 1;
end;
```

Двійковий пошук



X = 7



Вибрати середній елемент A[c] і порівняти з X.
Якщо X = A[c], знайшли (вихід).
Якщо X < A[c], шукати дальше в першій половині.
Якщо X > A[c], шукати дальше в другій половині.


```
nX := 0;
L := 1; R := N; {межі: шукаємо від A[1] до A[N] }
while R >= L do begin
  c := (R + L) div 2;
  if X = A[c] then begin
     nX := c;
     R := L - 1; { break; }
  end;
  if x < A[c] then R := c - 1;
  if x > A[c] then L := c + 1;
end;
if nX < 1 then writeln('Не знайшли...')
else          writeln('A[', nX, ']=', X);
```

номер середнього елемента

знайшли

вийти з циклу

зсуваємо межі

Способи заповнення масиву

```
const N = 5;
var a: array[1..N] of
integer;
  i: integer;
```

Оголошення масиву

```
for i:=1 to N do begin
 write('a[', i, ']=');
 read ( a[i] );end;
```

Заповнення оператором READ

```
for i:=1 to N do begin
a[i]=trunc(random(100));end;
```

Заповнення оператором RANDOM

```
writeln('Масив A:');
for i:=1 to N do write(a[i]:4);
```

Виведення масиву

Пошук максимального елементу

Задача: знайти в масиві максимальний елемент.

```
{ вважаємо, що перший елемент – максимальний }
for i:=2 to N do
 if a[i] > { максимального } then
   { запам'ятати новий максимальний елемент a[i] }
```

Додатково: як знайти номер максимального елемента?

```
max := a[1];                { вважаємо, що перший – максимальний }
iMax := 1;
for i:=2 to N do          { перевіряємо всі решта }
 if a[i] > max then       { знайшли новий максимальний }
 begin
   max := a[i];           { запам'ятати a[i] }
   iMax := i;           { запам'ятати i }
 end;
```

```
program qq;
const N = 5;
var a: array [1..N] of integer;
   i, iMax: integer;
begin
 writeln('Вихідний масив:');
 for i:=1 to N do begin
   a[i] := random(100) + 50;
   write(a[i]:4);
 end;
 iMax := 1; { вважаємо, що перший – максимальний }
 for i:=2 to N do    { перевіряємо всі решта }
   if a[i] > a[iMax] then { новий максимальний }
     iMax := i;          { запам'ятати i }
 writeln; {перейти на новий рядок}
 writeln('Максимальний елемент a[', iMax, ']=', a[iMax]);
end;
```

випадкові числа в інтервалі [50,150)

пошук максимального

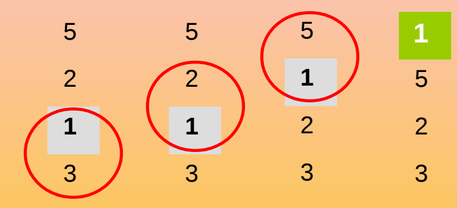
Сортування елементів масиву

Сортування – це розстановка елементів масиву в заданому порядку (по зростанню, спаданню, останній цифрі, сумі дільників, …).
Задача: переставити елементи масиву в порядку зростання.
Алгоритми: метод бульбашки, метод вибору

Метод бульбашки

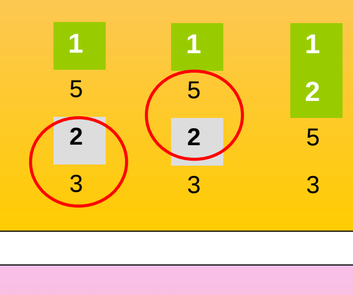
Ідея – бульбашка повітря в стакані води піднімається з дна вверх.
Для масивів – самий маленький ("легкий") елемент переміщується вверх ("спливає").

1-ий прохід



починаємо знизу, порівнюємо два сусідніх елементи; вони стоять "неправильно", міняємо їх місцями
за 1 прохід по масиву **один** елемент (самий маленький) стає на своє місце

2-ий прохід



3-ій прохід

Для сортування масиву з N елементів потрібен N-1 прохід (достатньо поставить на свої місця N-1 елемент).

```
program qq;
const N = 10;
var A: array[1..N] of integer;
   i, j, c: integer;
begin
 { заповнити масив }
 { вивести вихідний масив }
for i:=1 to N-1 do begin
   for j:=N-1 downto i do
     if A[j] > A[j+1] then begin
      c := A[j];
      A[j] := A[j+1];
     A[j+1] := c;
    end;
  end;
{ вивести одержаний масив }
end;
```