ELSŐ FEJEZET

Bevezetés az élet játékaiba

„Nekem a művészet a kommunikálás vágya."
(Szász Endre)

A számítógépvírusok kutatása sokak számára izgalmas terület, főleg olyanoknak, akik a természet, a biológia vagy épp a matematika iránt is érdeklődnek. Minden bizonnyal a számítógép-felhasználók többsége találkozott már a számítógépes vírusok egyre általánosabbá váló problémájával. Valójában a víruskutatók egy része csak akkor kezdett el komolyabban foglalkozni ezzel a témával, amikor évtizedekkel ezelőtt az ő számítógépüket is vírustámadás érte.

Donald Knuth könyvsorozatának[1] a címe, *A számítógép-programozás művészete,* arra utal, hogy bármi, ami egy számítógépnek megtanítható, az tudomány, és minden, a gép számára megtaníthatatlan dolog, művészet. A számítógépes vírusok kutatása gazdag, összetett és sokrétű tudomány, amely magába foglalja a kódvisszafejtést, valamint a felismerésre, hatástalanításra és védekezésre alkalmas rendszerek fejlesztését. Kétségtelenül vannak a víruskutatásnak tudományos aspektusai, jóllehet sok elemző módszer önmagában is művészet, nyilván ezért találja sok laikus nehezen érthetőnek ezt a viszonylag fiatal tudományterületet. Több év kutatási eredményei és publikációk sora után is számos olyan művészetnek számító elemzőtechnika létezik, amelyet csak a vírusölő programokat és a biztonsági rendszereket forgalmazó cégeknél lehet elsajátítani.

Ezen a területen sikereket azonban csak saját szervezet létrehozásával érhetünk el. Ez a könyv a víruskutatás érdekfeszítő világába enged betekintést, miközben – a szerző reményei szerint – olyan tudnivalókat mutat be, amelyek nemcsak a hivatásos informatikusok, hanem a fenti értelemben vett művészetet tanulók érdeklődését is felkelti. A cél a támadók és a rosszindulatú programok ellen védekező rendszerek felépítésének átfogó megismertetése.

Jóllehet számos könyv foglalkozik a számítógépes vírusok témakörével, mégis kevés olyan akad, amelynek szerzője kellő tapasztalattal rendelkezik a számítógépes vírusok kutatásában ahhoz, hogy a témát teljes körűen az érdeklődők elé tárja.

A következőkben az informatika történetének azokat a mozzanatait mutatjuk be, amelyek szerepet játszottak a számítógépes vírusok fejlődésében, majd a fejezet végén megadjuk a számítógépes vírus pontos definícióját.

1.1. Az önlemásoló struktúrák korai modelljei

A világot modellek segítségével különböző perspektívákból reprezentálhatjuk. Az önmagukat másoló struktúrákat modellező önreprodukáló rendszerek ötletét a magyar–amerikai Neumann János fogalmazta meg 1948-ban.[2, 3, 4]

Neumann matematikus volt, nagyszerű gondolkodó, és minden idők egyik legnagyobb számítógép-tervezője. A mai számítógépek az ő egykori elképzelése szerint épülnek fel. Az információ tárolásához bevezette a memóriát, és analóg helyett bináris műveletek használatát javasolta. Neumann öccse, Miklós szerint „Jancsira" nagy hatással volt Bach A *fúga művészete* című műve, amely több hangnemben íródott, és nincs határozott hangszerelése sem. Neumann Miklós egyenesen a tárolt programú számítógép ötletének forrásaként jelöli meg Bach művét.[5]

A hagyományos Neumann-gép alapvetően nem tesz különbséget kód és adat között. A kód akkor különül el az adattól, amikor az operációs rendszer átadta a vezérlést a kódnak, vagyis végrehajtotta a memóriában tárolt adatokat.

A későbbiekben látni fogjuk, hogy a biztonságos programozásnál alapvető, hogy minél inkább kézben tartsuk a programok és az adatok megkülönböztetését. Ugyanakkor nyilvánvaló, hogy ennek a módszernek hátrányai is vannak.

A modern számítógépek számos modellezési technikát alkalmaznak a természet szimulálására, ezek közül nem egy játék formájában jelenik meg. Bár a számítógépes vírusok sokban különböznek ezektől a természetmodellező játékoktól, a víruskutatást tanulók számára mégis segítséget jelenthetnek abban, hogy megértsék az önmagukat másoló struktúrákat.

1.1.1. Neumann János elmélete az önreprodukáló automatákról

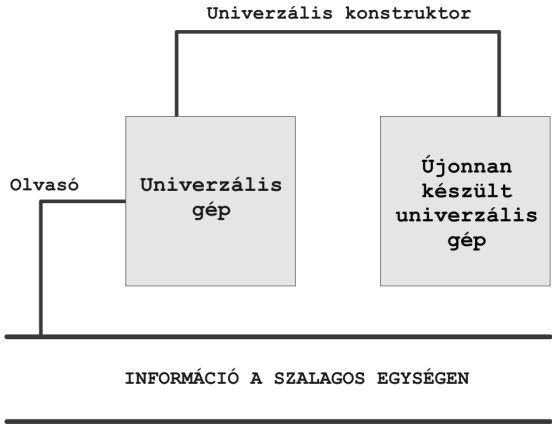
A szaporodás a természet egyik legalapvetőbb része. Neumann János elsőként javasolt olyan modellt, amelyben a természet önreprodukcióját egy önépítő automata ötletével írja le.

Neumann elképzelése szerint a rendszer három fő alkotóelemből áll:

1. egy univerzális gép,
2. egy univerzális konstruktor,
3. szalagon rögzített információ.

Az univerzális gép (Turing-gép) képes lenne olvasni a szalagot, és az azon lévő információ alapján az univerzális konstruktorral képes lenne lépésről lépésre újraalkotni önmagát. Maga a gép nem értené meg a folyamatot, csupán követni tudná a memóriaszalagon lévő információt (vázlatos utasításokat). Úgy lenne tehát felépítve, hogy csak a halmaz következő elemét érné el, és minden elemet egymás után megvizsgálna, amíg meg nem találná a megfelelőt. Ekkor két megfelelő darabot egymáshoz illesztene az utasításoknak megfelelően, és ezt a műveletsort ismételné mindaddig, amíg teljesen fel nem építette önmagát.

Amennyiben egy új rendszer felépítéséhez szükséges információ megtalálható lenne a szalagon, az automata képes lenne önmagát reprodukálni. A gép újraépülne *(1.1 ábra),* és az újonnan épített automata elindulásával ismét elkezdődhetne ugyanaz a folyamat.



1.1. ábra. *Egy önmagát újraépítő gép modellje*

Néhány évvel később Stanislaw Ulam azt javasolta Neumannak, hogy a modell leírására használja a cellás automatizálás módszerét. Ebben a gép részei helyett az egyes cellák (sejtek) állapotát írják le. Mivel a cellák robotként, szabályok („kód") szerint működnek, a cellát magát automatának nevezik. Ilyen cellák halmaza alkotja a sejtautomata- *(cellular automaton)* architektúrát (CA).

Neumann módosította az eredeti modellt, ebben a cellák 29 állapottal rendelkeztek egy kétdimenziós, ötcellás környezetben, és 200 000 cellát használt fel az önreprodukáló struktúrák elkészítéséhez. Ez a modell matematikailag bizonyította, hogy lehetséges az önreprodukáló struktúrák létrehozása. Szabályos élettelen részek (molekulák) kombinálhatók úgy, hogy önreprodukcióra képes struktúrák (potenciálisan élő organizmusok) jöjjenek létre.

Neumann 1948-ban publikálta az önreprodukáló automatákról szóló elméletét, mindössze öt évvel azelőtt, hogy 1953-ban Watson és Crick felfedezte, hogy az élő szervezetekben a DNS-lánc tartalmazza a szervezet felépítésére vonatkozó információkat, és az élőlények ezeket használják „memóriaszalagként" a szaporodáshoz.

Neumann János már nem élhette meg, hogy az elmélet bebizonyosodjon. Munkáját Arthur Burks fejezte be, majd 1968-ban E. F. Codd tökéletesítette. Codd leegyszerűsítette Neumann modelljét; mindössze nyolcállapotú cellákkal dolgozott ötcellás környezetben. Ilyen egyszerűsítés az alapja Christopher G. Langton és más mesterségesélet-kutatók munkájának is, amelynek során létrejöttek az úgynevezett önreprodukáló ciklusok.[6] Ezek a ciklusok a reprodukció szükségleteire koncentrálnak, miközben a bonyolult univerzális gépet eltüntetik a rendszerből.

Robert A. Freitas, Jr. és William B. Zachary[7] 1980-ban a NASA/ASEE-nél folytattak kutatásokat egy, a Holdon telepítendő, önreprodukáló és önállóan növekvő gyárral kapcsolatban. Kifejlesztettek egy olyan gyárépületet (LMF – lunar manufacturing facility), amely az önreprodukáló automaták elmélete és a létező automatizálási eljárások felhasználásával lehetővé tenné egy ilyen gyár telepítését a Holdon. Freitas, valamint Jr. és Ralph C. Merkle nemrégiben megjelent könyve, a *Kinematic Self-Replicating Machines* (Kinematikus önreprodukáló gépek) a téma iránti érdeklődés megújulását jelzi. Freitas néhány éve vezette be az ekofágia kifejezést, ezzel vázolja elméleti szinten a teljes ökoszisztéma összeomlását, amelyet elszabadult, önreprodukáló nanorobotok okoznak, majd katasztrófa mérséklésére tesz javaslatokat. [8]

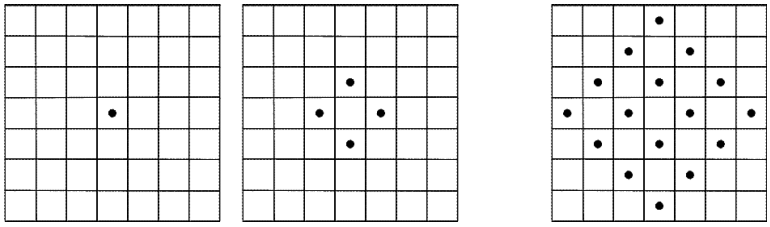
Az önreprodukáló gépek időről időre megjelennek a science fiction témáiban, a Terminátor-jellegű filmektől Neal Stephenson és William Gibson regényéig. Természetesen a science fiction világán kívül is találunk számos példát: a nanotechnológia és a mikroelektronikai mechanikus rendszerek (MEMS – microelectrical mechanical systems) tervezése ma már valódi tudomány.

1.1.2. Fredkin: szaporodó struktúrák

Neumann modelljét sokan próbálták leegyszerűsíteni. Edward Fredkin 1961-ben speciális sejtautomatát használt, amelyben egy rácson lévő összes struktúra képes volt reprodukálni magát egy egyszerű mintát követve (egy lehetséges megoldás látható az *1.2 ábrá*n). Fredkin automatája a következő szabályok szerint működött.[9]

- A táblán mindenütt ugyanolyan jel használatos.
- Minden lehetséges pozíciónak két állapota van: tartalmaz jelet, vagy nem.
- A jelek generációi véges időkereten belül követik egymást.
- Minden jel környezete meghatározza, hogy a következő generációban lesz-e az adott pozícióban jel.
- A jelek környezetét a felette, az alatta, a tőle jobbra és balra levő négyzetekben lévő jelek reprezentálják (ha az ötcellás Neumann-féle környezetet alkalmazzuk).

- Egy pozíció a következő generációban üres lesz, ha a környezetében páros számú jel van.
- Egy pozíción a következő generációban jel lesz, ha a környezetében páratlan számú jel van.
- Az állapotok száma változtatható.



1.2. ábra. *Első, második és negyedik generáció*

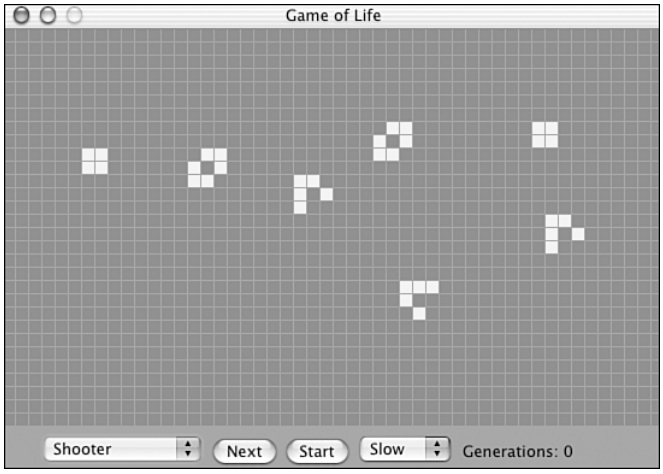
Ebből a kiinduló állásból, a fenti szabályokat felhasználva, minden struktúra reprodukálódik. Ennél nyilvánvalóan vannak érdekesebb elrendezések is, a fent bemutatott önreprodukáló sejtautomata a lehető legegyszerűbb modell.

1.1.3. Conway: „Élet” játék

Az egyik legérdekesebb sejtautomata rendszert John Horton Conway[10] készítette el 1970-ben. Az úttörőnek számító Neumannhoz hasonlóan ő is egyszerű elemek általános szabályrendszer szerinti kölcsönhatását vizsgálta, és azt tapasztalta, hogy ez meglepően érdekes szerkezeteket eredményezhet. Conway a játékát Életnek nevezte el, és a következő szabályokat használta:

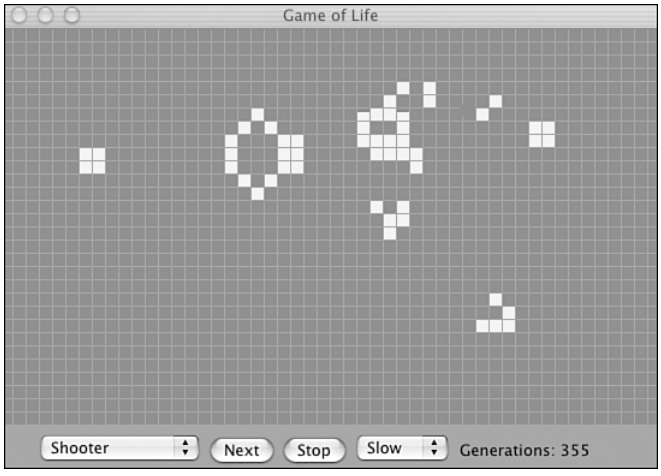
- Nem lehet olyan kezdőállapot, amely esetén bizonyítottan végtelenre nőne a populáció.
- Kell lennie olyan kezdőmintának, amely nyilvánvalóan végtelenül növekszik.
- A kezdőminták három genetikai törvény szerint változnak: születés, túlélés, halál.

Az *1.3. ábra* Conway játékának egy modern reprezentációját tartalmazza, amelyet Edwin Martin[11] készített el.



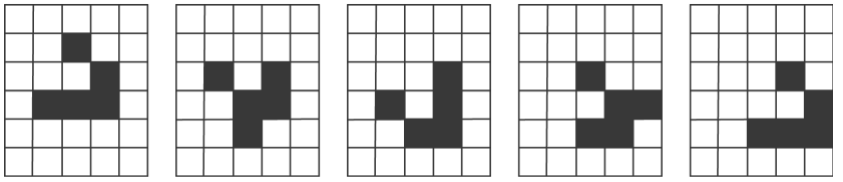
1.3. ábra. *Edwin Martin „Élet" játékának implementációja „Vadász" kezdőállapottal*

A „Vadász" kezdőállapottal indítva a program különösen érdekes számítógépes animációt hoz létre. Néhány generáció után a két egymás felé lövő vadász az *1.4 ábrá*nak megfelelően a tábla széleire kerül, és eközben sárkányrepülőket hoznak létre, amelyek a tábla jobb alsó sarka felé „repülnek" (lásd *1.5. ábra*). Ez a folyamat vég nélkül folytatódik, miközben egyre újabb sárkányrepülők jönnek létre.

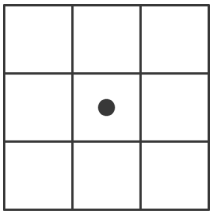


1.4. ábra. *A „Vadász" 355. generációja*

Egy kétdimenziós táblán egy cellának kétféle értéke lehet: S = 1, ha jel van a cellán, azaz a cella kitöltött, vagy S = 0, ha a cella üres. A cella élete a környezetétől függő szabályok szerint alakul (lásd *1.6. ábra*).



1.5. ábra. *A sárkányrepülők alakváltás nélkül mozognak*



1.6.ábra. *A kilenccellás alapú Moore-környezet*

Conway életjátékát a következő jellemzők/szabályok határozzák meg:

- **Születés:** Ha egy üres cellához három olyan szomszédos cella tartozik, amelyiken jel található (K = 3), akkor a cella a következő generációban kitöltött lesz.
- **Túlélés:** Ha egy kitöltött cellának két vagy három kitöltött szomszédja van (K = 2 vagy K = 3), akkor a cella az új generációban kitöltött marad.
- **A halál:** Ha egy kitöltött cellának csak egy vagy nulla kitöltött szomszédja van (K = 1 vagy K = 0), akkor a cella elszigeteltség miatt meghal. Ha egy cellának túl sok kitöltött szomszédja van: négy, öt, hat, hét vagy nyolc (K = 4, 5, 6, 7 vagy 8), akkor a cella a következő generációban túlnépesedés miatt meghal.

Conway eredetileg úgy gondolta, hogy az életjátékban nem fordulhatnak elő önreprodukáló struktúrák, ezért 50 dolláros jutalmat ajánlott fel annak, aki talál ilyen szerkezetet. Ám az MIT (Massachusetts Institute of Technology) mesterséges intelligenciával foglalkozó csoportja a számítógép segítségével rövidesen talált egy ilyen struktúrát.

Az MIT hallgatói fedezték fel azt a képződményt, amelyet később sárkányrepülőnek neveztek el. Tizenhárom sárkányrepülő összeérésekor létrejött egy olyan pulzáló objektum, amely a századik generációban újabb sárkányrepülőknek „adott életet", amelyek rögtön sebesen „elrepültek". Ezt követően minden harmincadik generációban újabb sárkányrepülő tűnik fel, és repül tovább a táblán. A sorozat végtelenül folytatódik, nagyon hasonlóan az *1.4.* és *1.3. ábrá*n látott „Vadász" elrendezéséhez.

Csákány Antal és Vajda Ferenc 1980-ban megjelent *Játékok számítógéppel* című könyvében vetélkedő játékokat írnak le. Ezek közül az egyik táblajáték szabályrendszere egészen hasonlít az életjáték szabályaihoz. A játékban az életért folytatott küzdelem a káposztá k, a nyulak és a rókák között zajlik. Egy kezdeti mezőn egy káposzta található, amely eledelül szolgál a nyúlnak, amelyet később pedig a róka eszik meg a meghatározott játékszabályok szerint. A szabályok meghatározzák és kiegyensúlyozzák a nyulak és a rókák számát a populációban.

Érdekesnek tűnik ebben az elgondolásban a számítógépek, a számítógépes vírusok és a vírusirtó programok kapcsolatát is megfigyelni. Számítógépek (pontosabban operációs rendszer, illetve valamilyen BIOS) nélkül a vírusok nem lennének képesek reprodukálódni. A szaporodó vírusok viszont egyre újabb számítógépeket fertőznek meg, mintegy követelve a vírusirtó programok létrejöttét.

Esetenként a vírusok visszavágnak; ezek az ún. *retrovírusok*. Ilyenkor a vírusirtó program „meghal". Ugyanakkor, ha a vírusirtó sikeresen semlegesít egy vírust, akkor természetesen a vírus „hal meg", illetve akadnak olyan szituációk is, amikor a vírusfertőzés pillanatában a számítógép azonnal „meghal".

Jó példa erre az, amikor egy vírus válogatás nélkül letörli az operációs rendszer kulcsfontosságú fájljait, és ezzel rendszerösszeomlást idéz elő, mondhatni „elpusztítja" a rendszert. Ha ez a folyamat túl gyorsan játszódik le, azaz a rendszer túlságosan hamar omlik össze, a vírusnak nincsen lehetősége a szaporodásra és újabb számítógépek megfertőzésére. Ha úgy tekintünk a sok millió számítógépre, mint a fenti táblajáték szereplőire, egyértelmű a tökéletes párhuzam a vírusok, a vírusirtó programok és a játékban modellezett káposzták, nyulak és rókák populációinak alakulása között.

Ezeknek a programoknak a végtelen harcában a szabályok, a mellékhatások, a mutációk és a fertőzőképesség foka teremti meg az egyensúlyt. Mindemellett a számítógépes vírusok és a vírusirtó programok között megfigyelhető egyfajta párhuzamos fejlődés is.[12] Minél kifinomultabbak lesznek a vírusirtók, annál fejlettebbé kell válniuk a vírusoknak is. Ez a tendencia a számítógépes vírusok 30 éves történetében folyamatosan megfigyelhető.

Az ilyen vonalak mentén felépített modellekből látható, hogyan változnak a vírusok párhuzamosan a velük kompatibilis számítógépekkel, illetve hogyan alakul a számítógépes vírusok és a vírusirtó programok egymással folyamatosan többszereplős vetélkedő játéka. Egy olyan környezetben, ahol sok kompatibilis számítógép található, a vírusok sokkal fertőzőbbek lesznek, és nagy sebességgel fertőznek meg újabb és újabb gépeket. Nagyszámú, kompatibilis operációs rendszerrel rendelkező PC olyan homogén környezetet hoz létre, amely termékeny talajt biztosít a vírusoknak (ugye ismerős?).

Kevesebb kompatibilis számítógépet reprezentáló kisebb táblán a kitörés is kisebb mértékű, és nyilvánvalóan viszonylag kisebb számú vírus is jelenik meg.

Ebből a modellből látszik, hogy miért jelenti a jelenlegi rendszerek 95%-át alkotó Windows a hatalmas „rács”-vírusfertőzésnek leginkább kitett részét; mindez természetesen nem jelenti azt, hogy a számítógépes rendszerek 5%-a nem képes globális járvány előidézésére.

> **Megjegyzés:** Az önreprodukáló, önjavító és fejlődő programok témakörében látogassunk el a http://lslwww.epfl.ch/biowall/index.html címre, és nézzük meg a BioWall projektet.

1.1.4. Core War: harci programok

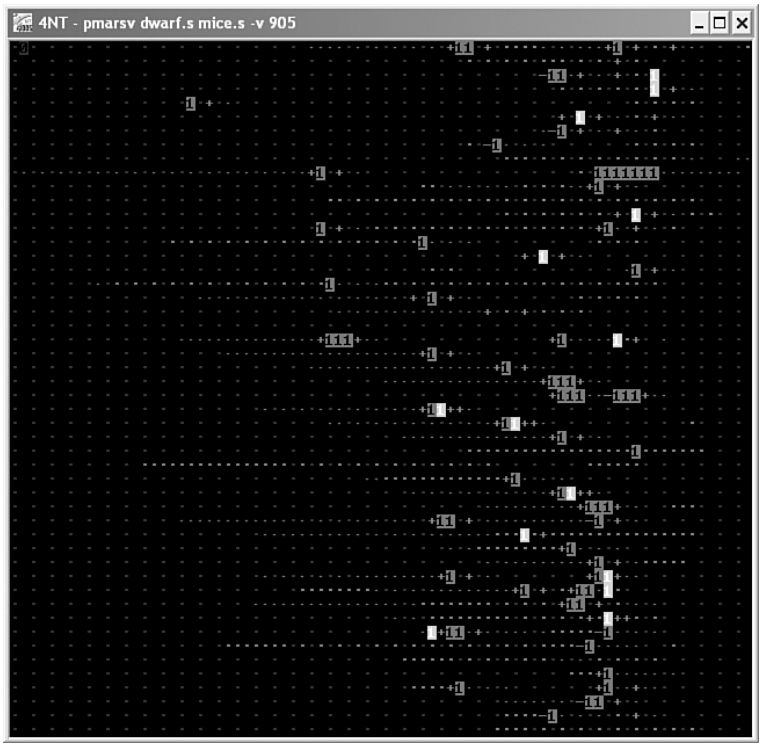
Robert Morris, később a Nemzetbiztonsági Hivatalnál vezető tudós, 1966-ban úgy döntött, hogy két barátjával létrehoz egy új játékkörnyezetet, amely aztán a *Darwin* nevet kapta. A program kódját két barát, Victor Vyssotsky és Dennis Ritchie készítette el. (Morris fia az a hírhedt vírusíró, aki a számítógépes vírusok történetében először készített féregvírust. A vírustörténelemben betöltött szerepéről a könyv későbbi részeiben még lesz szó.)

Az eredeti Darwin-programot a Bell laboratóriumban lévő PDP-1 gépre (programmed data processing – programozott adatfeldolgozás) írták, későbbi verziója lett Core War (magháború) elnevezésű számítógépes játék, amely a mai napig közkedvelt a programozók, a matematikusok (és a hekkerek) körében.

> **Megjegyzés:** A *hekker* kifejezést itt az eredeti, pozitív jelentésében használjuk: minden jó szándékú víruskutatót ebben a pozitív értelemben hekkernek tekintünk. A könyv szerzője ebben az értelemben maga is hekker, de alapvetően különbözik azoktól a romboló hekkerektől, akik mások számítógépét törik fel.

A játék azért kapta a Core War nevet, mert a játékosnak az a feladata, hogy az ellenfél programjainak felülírásával elpusztítsa azokat. A eredeti játékban két Redcode nyelven írt assembly program küzd egymás ellen. A programok egy szimulált (virtuális) gépen a MARS-on (Memory Array Redcode Simulator – memóriatömb Redcode szimulátor) futnak, és a két program harcát nevezik Core Warnak.

A Redcode eredeti utasításkészlete tíz egyszerű utasítást tartalmazott, amelyekkel információt lehetett átmozgatni egyik memóriaterületről a másikra. Ez a meglehetősen nagy flexibilitás miatt egészen trükkös harci programok írását tette lehetővé. Dewdney 1984 áprilisától több cikket is jelentetett meg a *Scientific American*[13,14] magazinban Számítógépes szórakozás címmel; ezek az írások a Core War játékot mutatták be. Az *1.7. ábrá*n a Core War játék PMARSV nevű implementációjából látható egy kép. Az implementációt Albert Ma, Sieben Nándor, Stefan Strack és Mintardjo Wangsaw készítette. Ebben a kis harcosok a MARS-környezeten belül küzdenek egymással.



1.7. ábra. *Core War harci programok (Dwarf és MICE) csatája*

A harci programok évenkénti csatájában a minden kihívóját legyőző program elnyeri a Hegy Királya (KotH – King of the Hill) címet. Az első ilyen versenyben a MICE program győzött, a fejlesztő, Chip Wendell jutalma pedig egy trófea volt, amelybe egy korai CDC 6600-as számítógép memóriakártyáját építették be.[14]

A legegyszerűbb Redcode program egyetlen MOV utasításból áll, amely az eredeti szintaxis szerint a következő: MOV 0,1. A program neve IMP, amely a 0 relatív címen lévő adatokat (amely jelen esetben maga a MOV, azaz a mozgatásoperátor) egy memóriacímmel hátrébb, azaz az 1-es relatív címre helyezi. Az utasítás végrehajtása után az új utasítás kapja meg a vezérlést, amely ismét eggyel magasabb címre másolja önmagát, és ez ugyanígy folytatódik tovább. Ez nyilvánvalóan azért történik így, mert mindig a következő címen lévő utasítás hajtódik végre legközelebb, miközben az utasításszámláló minden végrehajtott utasítással eggyel nő.

Az eredeti játékban 8000 cella állt rendelkezésre az utasítások számára, és egyszerre két program küzdhetett egymással. Az újabb verziókban már egyszerre több program is részt vehet a harcban. Kezdéskor minden programnak kötött számú, általában legfeljebb száz utasítása lehet, és a végrehajtott iterációk száma is limitált, ezt általában 80 000-ben szabják meg.

Az eredeti Redcode tíz utasításához képest az újabb verziók már nagyobb utasításkészlettel dolgoznak. Az *1.1. programlista* az 1994-es verzió 14 használható utasítását tartalmazza.

1.1. programlista. *Az 1994-es Core War-verzió utasításai*

```
DAT  data
MOV  move
ADD  add
SUB  subtract
MUL  multiply
DIV  divide
MOD  modula
JMP  jump
JMZ  jump if zero
JMN  jump if not zero
DJN  decrement, jump if not zero
CMP  compare
SLT  skip if less than
SPL  split execution
```

Tekintsük át a Dewdney Dwarf programjának oktatási segédletét (*1.2. programlista*).

1.2. programlista. *Dwarf bombázó harci program*

```
;name            Dwarf
;author          A. K. Dewdney
;version         94.1
;date April 29,  1993
;strategy        Bombs every fourth instruction.

ORG 1            ; Indicates execution begins with the second
                 ; instruction (ORG is not actually loaded, and is
                 ; therefore not counted as an instruction).

DAT.F   #0, #0   ; Pointer to target instruction.
ADD.AB  #4, $-1  ; Increments pointer by 4.
MOV.AB  #0, @-2  ; Bombs target instruction.
JMP.A   $-2, #0  ; Loops back two instructions.
```

A Dwarf úgynevezett bombázóstratégiát használ. Az első néhány sor megjegyzés, ahol a harci program neve, illetve az 1994-es Redcode-szabvány megjelölése látható. A Dwarf úgy győzi le az ellenfeleit, hogy DAT-„bombákat” „dob” az ellenfél utasításainak elérési útjára. Mivel egy program a MARS rendszerében azonnal meghal, ha megkísérel végrehajtani egy DAT utasítást, a Dwarf azonnal nyer, amint sikerül eltalálnia az ellenfelét.

A MOV utasítás arra szolgál, hogy a MARS celláiba adatokat vigyen be. (Az IMP-harcos ezt világosan megmagyarázza.) A Redcode-utasítások formátuma Utasításkód A, B, tehát a MOV.AB #0 @-2 utasítás forrásként a DAT utasítást jelöli meg.

Az A mező a DAT utasításra mutat, és mivel minden utasítás mérete 1, a 0 címen a DAT #0 #0 utasítást találjuk. Ebből következik, hogy a MOV utasítás a DAT utasítást fogja másolni arra a címre, amelyet a B mező ad meg. Meg kell tehát vizsgálnunk, hová mutat a B mező.

A B mező arra a címre mutat, ahol a DAT.F #0 #0 utasítás található. Ez általában azt jelenti, hogy ezt az utasítást felül kell írni a másolt utasítással, jelen esetben azonban a B mező elején található az @ jel, ez pedig azt jelenti, hogy ebben a mezőben indirekt címzést alkalmaztunk. Az @ jel jelenléte azt eredményezi, hogy a B által címzett helyen lévő utasítást újabb mutatóként, célcímként kell értelmezni. Ennek megfelelően a B mező a 0 helyet címzi (ahol a DAT.F utasítás található).

A MOV utasítás végrehajtása előtt azonban el kell végeznünk egy ADD utasítást. Az ADD #4 $-1 utasítás végrehajtása után a DAT-eltolás (ofszet) mezője néggyel nő, és ez az ADD utasítás minden végrehajtásakor bekövetkezik (így első lépésben 0-ból 4, majd 4-ből 8 lesz, és így tovább).

Ennek következtében, amikor a MOV utasítás ledobja a DAT-bombát, az mindig négy sorral (hellyel) a címzett DAT utasítás fölé kerül (*1.3. programlista*).

1.3. programlista. *A Dwarf kódja az első bomba ledobása után*

```
0     DAT.F #0, #8
1 ->  ADD.AB 4, $-1
2     MOV.AB #0, @-2  ; launcher
3     JMP.A $-2, #0
4     DAT ; Bomb 1
5     .
6     .
7     .
8     DAT ; Bomb 2
9     .
```

A JMP.A $-2 utasítás a vezérlést az aktuális pozíciónál kettővel előrébb lévő utasításnak, azaz az ADD-nak adja át, így a Dwarf-program futása „végtelenül" folytatódik. A program mindaddig rakosgatja le a bombákat minden negyedik pozícióba, amíg „körbeér" és visszatér. (A DAT-bombák utolsó lehetséges pozíciója után a program „körbeér" a memóriában, azaz visszatérve átlépi a nullát. Ha például az utolsó lehetséges hely a 10-es, akkor 10 + 1 értéke 0 lenne, 10 + 4 értéke pedig 3.)

Ekkor a Dwarf elkezdi sorban felülírni a saját bombáit mindaddig, amíg el nem éri a 80 000-es ciklushatárt (maximális iterációk száma), illetve amíg az ellenfél meg nem gátolja. A Dwarf-programot bármikor megsemmisítheti egy másik program, hiszen önmaga eltalálása, azaz a „baráti tűz" elkerülése érdekében a Dwarf végig egy helyben marad a memóriában, ezzel azonban minden más programmal szemben kiszolgáltatottá válik.

A Core War játékban a letapogatás, a szaporodás, a bombázás, az SPL-utasításokkal megvalósított IMP-spirál és a *Vámpír* néven ismert érdekes bombázási módszer általános stratégiának számít.

Dewdney arra is rámutatott, hogy egyes programok ellophatják az ellenfeleik „lelké”-t azzal, hogy eltérítik a végrehajtási menetüket. Ezek a programok az úgynevezett vámpír harcosok, akik JMP- (ugrás-) bombákat dobnak le a magban. Az ugrásbombákkal az ellenfél kódja eltéríthető egy olyan előre meghatározott pozícióba, ahol általában haszontalan kóddal fut tovább. Ez a haszontalan kód „felégeti” az ellenfél végrehajtási menetében a ciklusokat, nagy előnyhöz juttatva a vámpírprogramokat.

A vírusírás hasznos alternatívájaként ajánlhatók mindenkinek az ilyen veszélytelen, ugyanakkor érdekes játékok. A Core War új verziójában még a féregvírusok iránt érdeklődők sem fognak csalódni, hiszen ebben már különböző hálózatokon folyó csatákba is bekapcsolódhatunk, sőt akár át is lehet ugrani egyik csatából a másikba, ahol új ellenfelekkel mérhetjük össze a tudásunkat. A hálózati játék létrejöttével pedig akár féregszerű harci programok írására is lehetőség van.

1.2. A számítógép-vírusok születése

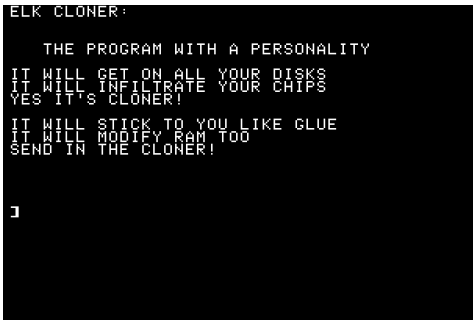
A vírusszerű programok az 1980-as években jelentek meg a mikroszámítógépeken. Meg kell említenünk két jelentős előfutárukat, a Creepert, amelyet 1971–72-ben készítettek, illetve John Walker 1975-ös programját, amely az UNIVAC gépre készült ANIMAL[15] játék „fertőző” verziója volt.

A Creeper, illetve nagy ellenfele a Reaper, a BBN PDP-10s gépein futó hálózati TENEX programja számára készült első „vírusirtó program” akkor született, amikor annak a projektnek a korai fejlesztései zajlottak, amelyet ma internetnek nevezünk.

Ennél még érdekesebb az ANIMAL program, amelyet egy UNIVAC 1100/42-es nagygépen fejlesztettek ki. és amely az UNIVAC 1100-as soros operációs rendszere, az Exec-8 alatt futott. John Walker (az Autodesk Inc. későbbi alapítója és az AutoCAD program egyik fejlesztője) 1975 januárjában írt egy olyan általános alprogramot, amelyet bármelyik program meghívhatott, magát az alprogramot PERVADE-nek[16] nevezte el. Amikor az ANIMAL meghívta a PERVADE-et, az végigment az összes elérhető könyvtáron, és az őt hívó programot, ez esetben az ANIMAL-t, bemásolta minden olyan könyvtárba, amelyhez a felhasználónak hozzáférése volt. Akkoriban még lyukszalagot használtak, így az adatcsere viszonylag lassan folyt, ennek ellenére az ANIMAL egy hónap múlva már számos helyen megjelent.

Az első mikroszámítógépes vírusok 1982 körül bukkantak fel az Apple-II gépeken. Az Elk Cloner nevű programot egy kilencedik osztályos pittsburghi fiú, Rich Skrenta[17] készítette, aki annak ellenére kezdett bele a fejlesztésbe, hogy maga sem hitte, hogy a programja működni fog. A barátai rendkívül szórakoztatónak találták az eredményt, a fiú matematikatanára, akinek a gépét megfertőzték a vírussal, már kevésbé. A fertőzött gép minden ötvenedik

újraindítása után az Elk Cloner betöltött egy képernyőt Skrenta versével *(1.8. ábra)*. Mivel a vírust csak a reset megnyomása indította be, az Elk Cloner minden ötvenedik betöltéskor maga kezdeményezte a gép újraindítását.



1.8. ábra. *Az Elk Cloner aktiválódik*

Skrenta több számítógépes játékot és számos hasznos programot is írt, és máig csodálkozik azon, hogy a hírnevét mégis élete „legidiótább kódjának" köszönheti.

1982-ben a Xerox PARC[18] két kutatója a számítógépes férgekről írt egy korai tanulmányt. Akkoriban ezekre a programokra még nem használták a *számítógépes vírus* kifejezést. A fogalmat 1984-ben vezette be Frederick Cohen,[19] akit korai tanulmányai miatt azóta is a számítógépes vírusok „atyjaként" tartanak számon. Cohen a *számítógépes vírus* elnevezést tanácsadója, Leonard Adleman professzor[20] javaslatára kezdte el használni. Maga a professzor science fiction regényekből vette a kifejezést.

1.3. Automatizált többszöröző kód: a számítógép-vírusok elmélete és definíciója

Cohen 1984-ben megalkotta a számítógépes vírusok formális matematikai modelljét, és ehhez egy Turing gépet használt. Valójában Cohen formális modellje nagyon hasonlít Neumann önreprodukáló sejtautomatákról készített modelljéhez. A számítógépes vírust úgy is tekinthetjük, mint egy Neumann-féle önreprodukáló sejtautomatát. Ennek a matematikai modellnek a mai kutatásokban nincsen gyakorlati szerepe, inkább csak általános leírást ad arról, mi is valójában a számítógépes vírus. Mindenesetre ez a matematikai modell jelentős elméleti alapot ad a számítógépes vírusok problémájához.

Cohen számítógépes vírusokról alkotott definíciója informálisan a következő: „A vírus olyan program, amely úgy fertőz meg más programokat, hogy azok kódját saját maga, esetleg fejlettebb másolatának beépítésével módosítja."

A vírus fontos tulajdonsága a fejlődés képessége, tehát az, hogy képes ugyanazt a kódot megváltoztatott formában lemásolni. A definíció a számítógépes vírus fontos jellemzőit foglalja össze, ugyanakkor a legszigorúbb értelemben véve félrevezető is.

Ezzel nem Cohen modelljét kívánjuk kritizálni, csupán jelezni szeretnénk, hogy rendkívül nehéz feladat a manapság létező rengeteg különféle vírus számára egyetlen pontos definíciót adni. Léteznek például olyan vírusok is, amelyek nem feltétlenül módosítják az áldozatként kiválasztott program kódját; ezek az úgynevezett *társvírusok* vagy *companion* vírusok. Ezek a programok nem igazodnak szigorúan Cohen definíciójához, hiszen nem másolják bele magukat más programok kódjába, hanem a programok operációs rendszer által tárolt környezetét változtatják meg oly módon, hogy saját magukat az áldozatprogram nevén belehelyezik a rendszer végrehajtási útvonalába. Ha egy viselkedésblokkoló program megírásakor szigorúan követjük Cohen informális definícióját, az így megírt, gyanús viselkedést figyelő blokkolóprogramok számára az ilyen vírusok problémát jelenthetnek, hiszen ha a definíció szerint megírt programok más programokat felülíró vírusokat keresnek, a társvírusokat a viselkedésük alapján nem fogják kiszűrni.

> **Megjegyzés:** Cohen matematikai formulája tökéletesen lefedi a társvírusokat is, a probléma csupán az egymondatos, köznyelvi megfogalmazás szó szerinti értelmezésénél adódik. A vírusok pontos definícióját egyetlen köznyelvi mondatban rendkívül nehéz meghatározni.

Az integritás-ellenőrző programok szintén arra alapoznak, hogy az egyes programok kódja időben változatlan. Ezek a rendszerek egy valamilyen kezdeti időpillanatban feltöltött adatbázis alapján dolgoznak, ezek az adatbázisok az egyes programok „tiszta" állapotát hivatottak tárolni. Cohen az integritás-ellenőrző programokat tartotta a vírusok elleni legjobb védekező módszernek, és ezzel a '90-es évek elején még egyet is lehetett érteni. Ugyanakkor egy társvírus ezeket a programokat is könnyedén kijátszhatja, ha az integritás-ellenőrző nem jelez a felhasználónak minden olyan alkalommal, amikor új alkalmazás kerül a gépre. Cohen saját rendszere ezt a technikát követte, ez határozottan a legbiztonságosabb módszer, jóllehet a felhasználóknak nem szükségszerűen tetszik, hogy a rendszer minden új program telepítésekor zavaró jelzéseket küld.

Cohen definíciója nem tesz különbséget az önmaguk másolására tervezett „valódi vírusok" és az olyan általános célú másolóprogramok (pl. fordítóprogramok) között, amelyek működésük mellékhatásaként képesek önmagukat is lemásolni.

A viselkedésblokkoló rendszerek a valóságban az ilyen programok esetén is veszélyt jeleznek. A népszerű parancsfeldolgozó, a Norton Commander, például használható arra, hogy saját kódját egy másik számítógépre vagy hálózatra másolja. Ez a művelet könnyen összetéveszthető egy önreprodukáló program működésével, különösen abban az esetben, ha a célkönyvtárban a program egy korábbi verziója található, amelyet a másolással kívánunk frissíteni. Ezek a „téves riasztások" könnyen kezelhetők, ugyanakkor kétségtelenül zavarók a végfelhasználók számára.

Ezeket a megjegyzéseket figyelembe véve a számítógépes vírus pontosabb definíciója a következő lehetne: „A számítógépes vírus olyan program, amely nyíltan és rekurzívan másolja saját maga, esetleg fejlettebb változatát."

Nincs szükség a másolás módjának pontos megadására, illetve egy másik alkalmazás vagy program szigorúan vett „megfertőzésére" vagy módosítására, jóllehet a legtöbb számítógépes vírus valóban úgy veszi át az irányítást, hogy módosít más programokat, ezért ezeknek a tevékenységeknek a blokkolása jelentősen lecsökkenti a rendszerben a számítógépes vírusok terjedésének a lehetőségét.

Ennek eredményeképp mindig van úgynevezett befogadó: ez egy olyan operációs rendszer vagy más futtatókörnyezet (pl. értelmezőprogram), amelyben jelek adott sorozata számítógépvírusként viselkedik, és rekurzívan többszörözi magát.

A számítógépes vírusok olyan önállósult programok, amelyek a felhasználó akarata ellenére lemásolják magukat, és új célpontok felé terjeszkednek. Egyes számítógépes vírusok ugyan feltesznek a felhasználónak egy „Biztosan meg akar fertőzni más programokat? (I/N)" típusú kérdést, de ezek ettől függetlenül még vírusok. A víruslaborok kezdő kutatói hajlamosak ezt másképpen értelmezni és amellett érvelni, hogy ezek a programok nem vírusok, ám ebben nyilvánvalóan tévednek.

Amikor az a kérdés, hogy egy programot a vírusokhoz sorolunk-e, alapvetően arra keressük a választ, hogy képes-e a program önmagát nyíltan és rekurzívan lemásolni. Egy olyan program, amelynek bármilyen segítségre van szüksége önmaga lemásolásához, nem tekinthető vírusnak. A program környezetének megváltoztatása (akár a lemezen lévő adatok kézi átírása) vagy akár a vírusnak szánt program kódjának hibakeresővel történő kijavítása ilyen segítségnek minősül. Az efféle nem működő vírusokat *tervezett vírusoknak* nevezzük.

A másolat nem szükségszerűen azonos az eredeti példánnyal, hiszen a modern vírusokra, különösen az úgynevezett metamorf vírusokra (részletesen lásd a 7. fejezetben) jellemző, hogy olyan másolatokat készítenek önmagukról, amelyeknek a kódja az egyes generációkban tökéletesen különbözik az eredetitől, működésükben mégis vele azonos vagy hasonló lesz.

Hivatkozások

1. Knuth, Donald E.: *The Art of Computer Programming*. 2nd Edition, Addison–Wesley, Reading, MA, 1973, 1968.
2. von Neumann, John: *The General and Logical Theory of Automata*. Hixon Symposium, 1948.
3. von Neumann, John: *Theory and Organization of Complicated Automata*. Lectures at the University of Illinois, 1949.
4. von Neumann, John: *The Theory of Automata: Contruction, Reproduction, Homogenity*. Unfinished manuscript, 1953.
5. Poundstone, William: *Prisoner's Dilemma, Doubleday*. New York, 1992.
6. Bachmutsky, Eli: *Self-Replication Loops in Cellular Space*. http://necsi.org:16080/postdocs/sayama/sdrs/java.
7. Freitas, Jr., Robert A.–Zachary, William B.: A *Self-Replicating, Growing Lunar Factory*. Fifth Princeton/AIAA Conference, May 1981.
8. Freitas, Jr., Robert A.: *Some Limits to Global Ecophagy by Biovorous Nanoreplicators, with Public Policy Recommendations*. http://www.foresight.org/nanorev/ecophagy.html.
9. Marx, György: A *természet játékai*. Ifjúsági Lap és Könyvterjesztő Vállalat, Hungary, 1982.
10. Gardner, Martin: *Mathematical Games: The Fantastic Combinations of John Conway's New Solitaire Game 'Life'*. Scientific American, October 1970, 120–123.
11. Martin, Edwin: *John Conway's Game of Life*. http://www.bitstorm.org/gameoflife.
12. Nachenberg, Carey: *Computer Virus-Antivirus Coevolution*. Communications of the ACM, January 1997, Vol. 40, No. 1., 46–51.
13. Dewdney, A. K.: *The Armchair Universe: An Exploration of Computer Worlds*. New York: W. H. Freeman (c), 1988.
14. Dewdney, A. K.: *The Magic Machine: A Handbook of Computer Sorcery*. New York: W. H. Freeman (c), 1990.
15. Walker, John: *ANIMAL*. http://fourmilab.ch/documents/univac/animal.html.
16. Walker, John: *PERVADE*. http://fourmillab.ch/documents/univac/pervade.html.
17. Skrenta, Rich: http://www.skrenta.com.

18. Shock, John–Hepps, Jon: *The Worm Programs, Early Experience with a Distributed Computation*. ACM, Volume 25, 1982, 172–180.

19. Cohen, Frederick B.: *A Short Course on Computer Viruses, Wiley Professional Computing*. 2nd edition, New York, 1994.

20. Bontchev, Vesselin Vladimirov: *Methodology of Computer Anti-Virus Research*. University of Hamburg Dissertation, 1998.