

# Teoretyczne podstawy informatyki



## Wykład 5a: Modele danych – wprowadzenie

<http://hibiscus.if.uj.edu.pl/~erichter/Dydaktyka2009/TPI-2009>

# Abstrakcja

---

## □ Abstrakcja:

- Oznacza uproszczenie, zastąpienie skomplikowanych i szczegółowych okoliczności występujących w świecie rzeczywistym zrozumiałym modelem umożliwiającym rozwiązanie naszego problemu.
- Oznacza to, że „abstrahujemy” od szczegółów, które nie mają wpływu lub mają minimalny wpływ na rozwiązanie problemu.
- Opracowanie odpowiedniego modelu umożliwia zajęcie się istotą problemu.

# Modele danych

---

- **Modele danych są to abstrakcje wykorzystywane do opisywania problemów.**
- Każdą koncepcję matematyczną można opisać za pomocą modelu danych. W informatyce wyróżniamy zazwyczaj dwa aspekty:
  - Wartości które nasz obiekt może przyjmować.  
Przykładowo wiele modeli danych zawiera obiekty przechowujące wartości całkowitoliczbowe. Ten aspekt modelu jest **statyczny**; określa bowiem wyłącznie grupę wartości przyjmowanych przez obiekt.
  - Operacje na danych.  
Przykładowo stosujemy zazwyczaj operacje dodawania liczb całkowitych. Ten aspekt modelu nazywamy **dynamicznym**; określa bowiem metody wykorzystywane do operowania wartościami oraz tworzenia nowych wartości.
- **Badanie modeli danych, ich właściwości oraz sposobów właściwego ich wykorzystania stanowi jedno z podstawowych zagadnień informatyki.**

## Modele danych a struktury danych

---

- **Modele danych to abstrakcje** wykorzystywane do opisywania problemów.
- **Struktury danych to reprezentacja danego modelu danych**, którą musimy skonstruować w sytuacji gdy język programowania nie ma wbudowanej tej reprezentacji.
- **Konstruujemy strukturę danych za pomocą abstrakcji obsługiwanych przez ten język.**

# Modele danych języków programowania

---

- Każdy język programowania zawiera własny model danych, który zazwyczaj istotnie różni się od modeli oferowanych przez inne języki.
- Podstawowa zasada realizowana przez większość języków programowania w odniesieniu do modeli danych określa, że każdy program ma dostęp do „pudełek”, które traktujemy jako obszary pamięci.
  - Każde „pudełko” ma swój typ, np. int, char.
  - Wartości przechowywane w pudełkach nazywamy często obiektami danych.
  - Możemy teraz nadawać nazwy wykorzystywanym pudełkom. W ogólności nazwa jest dowolnym wyrażeniem wskazującym na pudełko.

# Modele danych języków programowania

---

- Podstawowe typy danych w języku programowania **C** to:
  - liczby całkowite,
  - liczby zmiennoprzecinkowe,
  - znaki,
  - tablice,
  - struktury,
  - wskaźniki.
- Wszystkie te pojęcia to **statyczne elementy modelu danych**.
- Dopuszczalne operacje na tych danych to:
  - typowe operacje arytmetyczne na liczbach całkowitych i zmiennoprzecinkowych,
  - operacje dostępu do elementów tablic i struktur,
  - oraz wyluskiwanie wskaźników czyli znajdowanie obiektów przez nie wskazywanych.
- Te operacje to **dynamiczne elementy modelu danych**.

# Modele danych języków programowania

---

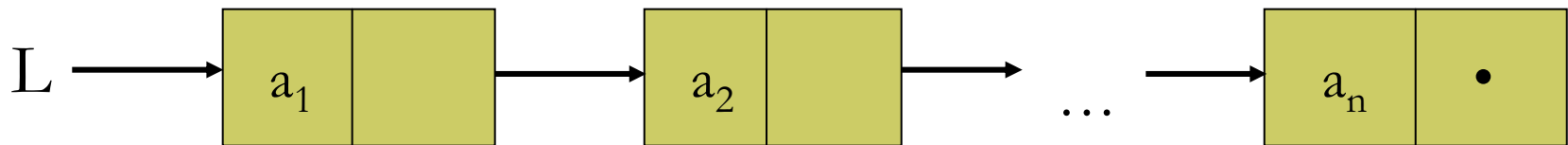
- Bardzo ważne są też **modele danych**, które **nie są** częścią języka programowania, takie jak listy, drzewa, grafy, zbiory.
- Np. w języku matematycznym, lista jest ciągiem  **$n$**  elementów, który zapisujemy jako  **$(a_1, a_2, \dots, a_n)$** .  
Do zbioru operacji wykonywanych na listach należą:
  - tworzenie listy,
  - wstawianie nowego elementu do listy,
  - usuwanie elementu z listy,
  - łączenie list.

# Modele danych a struktury danych

- Lista jest to abstrakcja matematyczna lub model danych.
- Lista jednokierunkowa to struktura danych:

```
typedef struct CELL *LIST;  
struct CELL{  
    int element;  
    LIST next;  
}
```

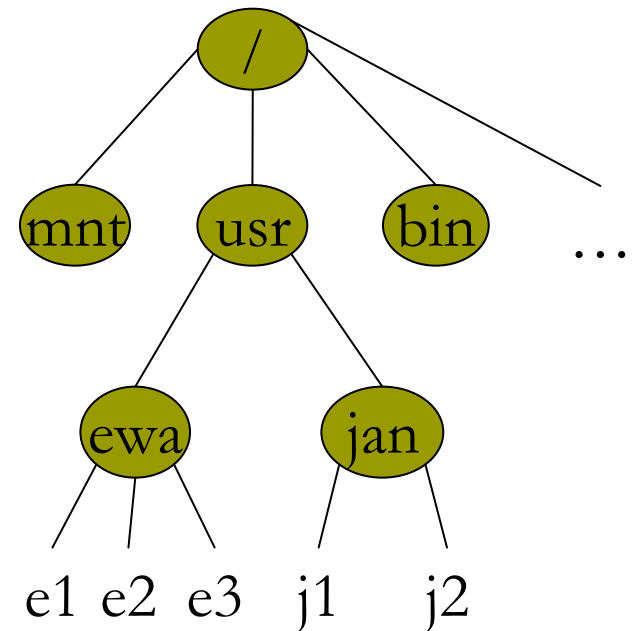
W niektórych językach (Lisp, Prolog) nie ma potrzeby stosowania (konstruowania) struktur danych do reprezentowania abstrakcyjnych list.





## Modele danych w oprogramowaniu systemowym

- Modele danych możemy też spotkać w systemach operacyjnych i w aplikacjach. Zadaniem systemu operacyjnego jest zarządzanie i szeregowanie zasobów komputera. Model danych systemów operacyjnych Unix składa się z takich pojęć jak **pliki**, **katalogi** oraz **procesy**.
  - Dane jako takie są przechowywane w **plikach** (ang. files), które w systemie Unix reprezentowane są przez ciągi znaków.
  - Pliki są grupowane w ramach **katalogów** (ang. directories), będących zbiorami plików i (lub) innych katalogów.
  - Katalogi i pliki tworzą **drzewo** w którym pliki są liśćmi.



## Modele danych w oprogramowaniu systemowym

---

- **Procesy** są pojedynczymi wykonaniami programów. Procesy pobierają zero lub więcej strumieni wejściowych i produkują zero lub więcej strumieni wyjściowych. W systemach Unix procesy mogą składać się z **potoków** (ang. pipes), kiedy to wynik jednego procesu może zasilać wejście kolejnego procesu. Efekt takiego połączenia procesów można traktować jako jeden duży proces z własnym wejściem i wyjściem.

- Przykład:

```
ls | grep file
```

- Istnieje wiele innych aspektów działania systemu operacyjnego, np. sposób zarządzania bezpieczeństwem danych oraz interakcja z użytkownikiem.
- **Dość łatwo można zauważyć że model danych systemu operacyjnego różni się od modeli danych języków programowania.**

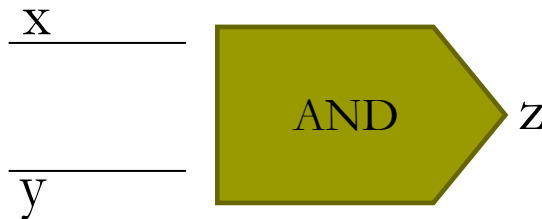
## Model danych w edytorach tekstu

---

- Każdy model danych wbudowany w taki edytor wiąże się z pojęciami ciągów tekstowych oraz operacjami charakterystycznymi dla redagowania tekstu.
- Model zawiera więc zazwyczaj pojęcie **wierszy** (ang. lines), które podobnie jak większość plików są ciągami znaków. Jednak w przeciwieństwie do plików wiersze mogą się wiązać ze swoimi numerami. Mogą być także grupowane w większe jednostki zwane **akapitami**.
- **Operacje na wierszach** można zazwyczaj stosować dla wszystkich zawartych w nich elementów, nie tylko dla ich początku, jak w przypadku najbardziej powszechnych operacji na plikach.
- Typowy edytor wykorzystuje również pojęcie wiersza bieżącego oraz bieżącej pozycji w danym wierszu. Wykonywane przez edytor operacje zawierają rozmaite modyfikacje wierszy, takie jak usuwanie i wstawianie znaków, usuwanie lub tworzenie nowych wierszy, poszukiwanie określonych ciągów znaków, itd.

# Modele danych układów komputerowych

- ❑ Model danych opisujący układy komputerowe, zwany **logiką wnioskowania**, jest najbardziej przydatnym narzędziem w projektowaniu komputerów.
- ❑ Komputery składają się z komponentów elementarnych zwanych bramkami (ang. gates). Każda bramka ma jedno lub więcej wejść i jedno wyjście; na wejściu i wyjściu dopuszczalne są tylko dwie wartości: **0** lub **1**. Bramka wykonuje prostą funkcję – np. koniunkcję (bramka **AND**).
- ❑ Na pewnym poziomie abstrakcji projektowanie komputera jest procesem, w którym decyduje się o sposobie połączenia bramek tak, by możliwe było efektywne wykonywanie na nim prostych operacji.



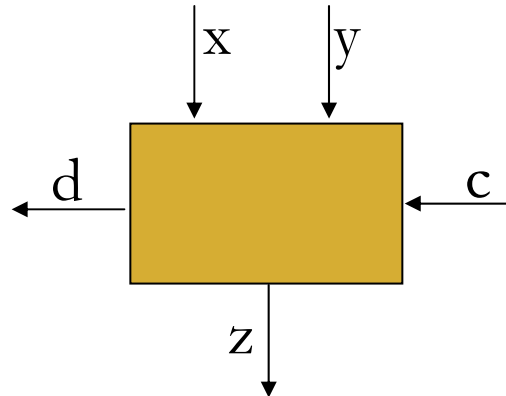
x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

# Sumator jednobitowy

- Aby wykonać instrukcję przypisania  $a = b + c$  w języku **C**, komputer wykonuje dodawanie za pomocą układu zwanego sumatorem (ang. adder). W komputerze wszystkie liczby są zapisywane w notacji binarnej wykorzystującej dwie cyfry, **0** i **1** (zwane **cyframi binarnymi** lub **bitami**). Mając kilka bramek możemy zbudować układ zwany **sumatorem jednobitowym** (ang. one bit adder). Dwa bity wejściowe, **x** i **y**, oraz wejściowy bit przeniesienia, **c**, są sumowane. Efektem tej operacji jest bit sumy oraz wyjściowy bit przeniesienia (ang. carry out) **d**.

Tabela prawdy:

- Przykład:
  - **dz** to łącznie dwubitowa liczba binarna wyrażający łączną liczbę danych wejściowych (**x**, **y**, **c**) mających wartość 1,
  - **d** = bit przeniesienia,
  - **z** = bit sumy.



x	y	c	d	z
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
1	0	0	0	1
0	1	1	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

## Modele danych języka C

---

- Statyczna część modelu danych w języku C to system typów opisujący wartości, które mogą być przyjmowane przez określone dane.
- System typów zawiera **typy proste**, np. liczby całkowite, oraz zbiór **zasad formowania typów**, dzięki którym możemy konstruować coraz bardziej skomplikowane typy na bazie typów już znanych.
- **Typy podstawowe:**
  - znaki (char, signed char, unsigned char),
  - liczby całkowite (int, short, long int, unsigned),
  - liczby zmiennoprzecinkowe (float, double, long double),
  - Wyliczenia (enum).
- Liczby całkowite i zmiennoprzecinkowe traktowane są jako typy arytmetyczne.

## Modele danych języka C

---

- Reguły formowania typów wymagają istnienia pewnych typów które mogą być albo typami podstawowymi; albo typami wcześniej skonstruowanymi za pomocą takich reguł.
- **Typy tablicowe:**
  - Możemy stworzyć tablice, której elementy są typu **T**: **T A[n]**;
  - Powyższa instrukcja deklaruje tablice **n** elementów, każdy typu **T**.
  - W języku **C** indeksy tablic rozpoczynają się od **0**, zatem pierwszym elementem jest **A[0]**, ostatnim **A[n-1]**.
  - Tablice mogą być skonstruowane ze znaków, typów arytmetycznych, wskaźników, struktur, unii lub innych tablic.

# Modele danych języka C

## □ Struktury:

- Struktura jest grupowaniem zmiennych zwanych **składnikami** (ang. members) lub **polami** (ang. fields). Różne składniki struktur mogą być różnych typów, jednak każdy musi zawierać elementy jednego określonego typu.
- Jeśli  $T_1, T_2, \dots, T_n$  są typami oraz  $M_1, M_2, \dots, M_n$  są nazwami składników, to deklaracja:

```
struct S {  
    T1 M1;  
    ....  
    Tn Mn;  
}
```

definiuje strukturę której wyróżnik (nazwa jej typu) to S, zaś **n** to liczba jej składników, **i**-ty składnik nosi nazwę **M<sub>i</sub>** i jest typu **T<sub>i</sub>**.

- Wyróżnik struktury S jest opcjonalny, udostępnia jednak wygodny skrót podczas odwoływania się do tego typu w późniejszych deklaracjach.



## Modele danych języka C

### □ Unie:

- Unia pozwala na przechowywanie zmiennych przyjmujących wartości różnych typów w różnych momentach wykonywania programu.
- Deklaracja:

```
union {  
    T1 M1;  
    T2 M2;  
    ...  
    Tn Mn;  
}
```

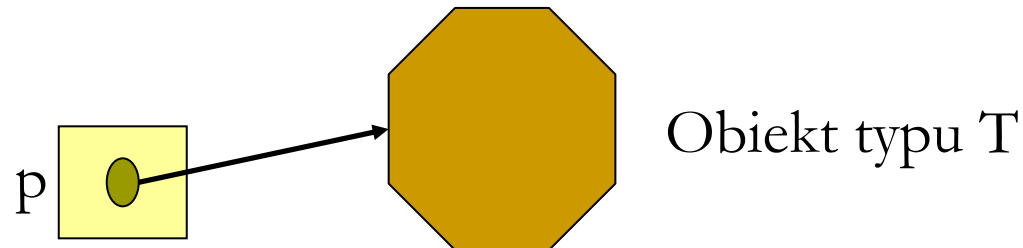
**x**;

definiuje zmienną **x**, która może przechowywać wartość dowolnego typu z grupy **T**<sub>1</sub>, **T**<sub>2</sub>, ..., **T**<sub>n</sub>. Nazwy składników **M**<sub>1</sub>, **M**<sub>2</sub>, ..., **M**<sub>n</sub> pomagają wyróżnić typ aktualnej wartości zmiennej. Oznacza to że **x.M**<sub>i</sub> wskazuje na wartość zmiennej **x** traktowanej jako wartość typu **T**<sub>i</sub>.

## Modele danych języka C

### □ Wskaźniki:

- Język **C** wyróżnia się znaczeniem jaki mają w nim wskaźniki. Zmienna typu wskaźnikowego zawiera adres obszaru pamięci. Za pomocą wskaźnika możemy uzyskać dostęp do wartości innej zmiennej.
- Deklaracja: **T \*p;**  
definiuje zmienną **p**, jako wskaźnik do zmiennej typu **T**.
- Zmienna **p** nazywa więc pudełko typu wskaźnikowego do **T**, wartością w pudełku **p** jest wskaźnik. Tym co „naprawdę” znajduje się w pudełku jest adres (lokacja), pod którym obiekt typu **T** jest przechowywany w komputerze.



# Modele danych języka C

- **Typedef:**
  - Język **C** udostępnia instrukcje **typedef**, która umożliwia tworzenie synonimów dla nazw typów.
  - Deklaracja: **typedef int Odległość**; pozwala na późniejsze używanie nazwy **Odległość** zamiast typu **int**.
- **Funkcje:**
  - Funkcje także posiadają związane ze sobą typy, mimo że nie łączymy z nimi pudełek ani wartości.
  - Dla dowolnej listy typów **T<sub>1</sub>, T<sub>2</sub>, ..., T<sub>n</sub>** możemy zdefiniować funkcje pobierającą odpowiednio **n** parametrów tych typów. Typ wartości zwracanych przez funkcje nazywamy typem funkcji. Jeżeli funkcja nie zwraca żadnej wartości wykorzystujemy typ **void**.
- W ogólności możemy budować typy dowolnie, stosując reguły ich konstrukcji, istnieje jednak kilka ograniczeń. Przykładowo nie możemy konstruować tablicy funkcji mimo że możemy zbudować tablice wskaźników do funkcji.

# Operacje w modelu danych języka C

---

- Przewidywane operacje na danych w modelu języka C możemy podzielić na trzy kategorie:
  - operacje tworzące i usuwające obiekt danych,
  - operacje dostępu i modyfikacji części obiektu danych,
  - operacje łączące części obiektu danych w celu sformowania nowej wartości obiektu danych.

# Operacje w modelu danych języka C

---

## □ Tworzenie i usuwanie obiektu danych:

- Język **C** udostępnia wiele elementarnych mechanizmów przeznaczonych do tworzenia danych.
- W momencie wywołania funkcji tworzone są pudełka dla wszystkich jej lokalnych argumentów (parametrów). Pozwala to na przechowywanie wartości tych parametrów.
- Innym mechanizmem jest procedura biblioteczna **malloc(n)**, która zwraca wskaźnik do **n** kolejnych pozycji znaków w niewykorzystanej pamięci. Obiekty danych mogą być wówczas utworzone właśnie w tych obszarach pamięci.
- Metody usuwania obiektów danych są analogiczne. Procedura biblioteczna **free** zwalnia pamięć zarezerwowaną przez **malloc**.

# Operacje w modelu danych języka C

## □ Dostęp do danych i ich modyfikacja:

- Język **C** zawiera mechanizm umożliwiający dostęp do komponentów składających się na obiekty. Wykorzystujemy:
  - zapis **a[i]** do uzyskania dostępu do **i-tego** elementu tablicy **a**,
  - zapis **x.m** do uzyskania dostępu do składnika **m** struktury o nazwie **x**
  - zapis **\*p** do uzyskania dostępu do obiektu wskazanego przez wskaźnik **p**.
- Modyfikowanie (przypisywanie) wartości w języku **C** realizujemy za pomocą operatorów przypisania, które umożliwiają zmianę wartości obiektu. Np.:
  - **a[0].(\*pole[3]) = 99.**

# Łączenie danych

- Język **C** zawiera bogaty zbiór operatorów umożliwiających manipulowanie danymi i łączenie ich wartości. Oto podstawowe operatory:
  - **Operatory arytmetyczne:**
    - dwuargumentowe  $+$ ,  $-$ ,  $\cdot$ ,  $/$ ,  $\%$ ,
      - $3 + 2 = 5$ ;
      - $3 - 2 = 1$ ;
      - $3 \cdot 2 = 6$ ;
      - $3 / 2 = 1.5$ ;
      - $3 \% 2 = 1$  (modulo – reszta z dzielenia);
    - jednoargumentowe  $+$ ,  $-$ ,  $++$ ,  $--$ ,
      - $n = -k$  ( $-$  jako zmiana znaku liczby);
      - $n++$ ,  $++n$  (inkrementacja, zwiększanie wartości o 1);
      - $n--$ ,  $--n$  (dekrementacja, zmniejszenie wartości o 1);

# Łączenie danych

## ■ Operatory logiczne:

Język **C** nie zawiera typu **Boolean**, wykorzystuje **0** do reprezentowania wartości logicznej fałszu oraz liczby różnej od zera do reprezentowania prawdy.

Język **C** udostępnia:

- koniunkcje **&&** (dwuargumentowy)
- alternatywę **||** (dwuargumentowy)
- negacje **!** (jednoargumentowy)
- operator warunkowy **warunek ? y : z** (trzyargumentowy), znaczący:  
if (**warunek**)  
    then return **y**;  
    else return **x**;



# Łączenie danych

---

- **Operatory porównania:** (`==`, `!=`, `<`, `>`, `≤`, `≥`)
  - Dla liczb całkowitych i zmiennoprzecinkowych. Wynikiem jest prawda lub fałsz.
- **Operatory działań na poziomie bitowym**
- **Operatory przypisania**
- **Operatory koercji (konwersja, rzutowanie):**
  - przekształcenie wartości jednego typu na odpowiadającą jej wartość innego typu.
- Często uzupełnia się podstawowe typy przez identyfikatory zdefiniowane w pliku nagłówkowym `stdio.h`: **NULL**, **TRUE**, **FALSE**, **BOOLEAN**, **EOF**.

# Bazy danych

---

- ❑ W wielu zastosowaniach komputerów same struktury danych nie wystarczają. Nie zawsze jest to bowiem tylko kwestia rozważenia zadania algorytmicznego i zdefiniowania dobrych i użytecznych do jego rozwiązania struktur danych. Czasem potrzeba bardzo obszernych zasobów danych, stanowiących dla wielu algorytmów potencjalne dane wejściowe, a więc mające ustaloną strukturę i nadające się do odszukiwania i manipulowania nimi.
- ❑ Przykładami takich danych mogą być finansowe i osobowe dane przedsiębiorstwa, rezerwacje miejsc i informacje o lotach towarzystwa lotniczego, dane katalogowe biblioteki, itd....

# Bazy danych

---

- ❑ Bazy danych są zazwyczaj bardzo obszerne i zawierają wiele różnych rodzajów danych, począwszy od nazwisk i adresów po specjalne kody i symbole, a czasami nawet zwykły tekst.
- ❑ Zgromadzone dane są zazwyczaj przedmiotem licznych rodzajów operacji wstawiania, usuwania i wyszukiwania, wykorzystywanych w różnych celach przez różnych ludzi.
- ❑ O ile dodanie nowej informacji do bazy danych lub usunięcie już istniejącej są zadaniami stosunkowo łatwymi, o tyle **zapytanie** bazy danych z zamiarem wydobywania z niej informacji zazwyczaj jest dużo bardziej skomplikowane.

# Bazy danych

---

- ❑ Ogromne znaczenie ma dobra organizacja bazy danych.
- ❑ Tak jak w przypadku struktur danych, dobry projekt bazy danych – to projekt przejrzysty, łatwy do zapisania, najważniejsza zaleta to duża sprawność działania i wykonalność opartego na tym projekcie systemu zarządzania bazą danych który potrafi odpowiedzieć na zapytania w krótkim czasie.
- ❑ Stosuje się różne modele organizacji baz danych.
- ❑ Modele, zaprojektowane do obsługi dużych ilości danych, jednocześnie wiernie i sprawnie wychwytyją związki zachodzące między obiektami danych.
- ❑ Istnieje wiele metod i języków manipulacji danymi i zapytań baz danych.

# Bazy danych

---

- Jeden z najpopularniejszych, **model relacyjny**, zaspokaja potrzeby związane z układami danych w postaci ogromnych tabel, przypominających tablicowe struktury danych.
- Inny model, **model hierarchiczny**, wymaga pewnych rodzajów układów drzewiastych albo sieciowych. Ten model organizuje dane w drzewiastej formie o wielu poziomach.
- Na niektóre rodzaje danych lepiej patrzeć jak na fragmenty **wiedzy** niż tylko jako na liczby, nazwiska czy kody.
- Oprócz dużej bazy danych opisującej inwentarz przedsiębiorstwa produkcyjnego moglibyśmy chcieć mieć dużą bazę informacji dotyczących prowadzenia tego przedsiębiorstwa. Tego rodzaju fragmenty wiedzy wymagają organizacji bardziej złożonej niż obiekty danych o mniej więcej ustalonym formacie, zwłaszcza wówczas gdy zależy nam na sprawnym wyszukiwaniu.

## Bazy wiedzy

---

- **Bazy wiedzy** stają się następnym naturalnym stopniem po **bazach danych**, są bogatym źródłem ciekawych pytań związanych z reprezentowaniem, organizacją i wyszukiwaniem algorytmicznym.
- Problem **reprezentacji wiedzy** jest faktycznie jednym z podstawowych zagadnień **sztucznej inteligencji**.
- Trudność wynika z tego, że wiedza składa się nie tylko z wielkiego zbioru faktów, ale także wielu zawiłych związków między nimi. Te związki implikują inne, wyższego poziomu związki z innymi elementami wiedzy.

## Bazy wiedzy

---

- Zaproponowano wiele **modeli wiedzy** które można by wykorzystać w inteligentnych programach. Niektóre opierają się na pojęciach czysto informatycznych, takich jak relacyjne czy hierarchiczne bazy danych. Inne na logicznych formalizmach takich jak rachunek predykatów czy logika modalna.
- Pewne języki programowania, jak **Lisp** czy **Prolog**, łatwiej nadają się do manipulowania wiedzą niż inne.
- Np. **Prolog** wydaje się trafnie dobrany jeżeli chodzi o fragmenty wiedzy dotyczące prostych relacji.
- Wymagane związki stają się coraz bardziej zagnatwane gdy wyjdziemy poza małą dobrze określoną dziedzinę dyskusji. Sięganie do wiedzy wiążącej się z pewną decyzją, którą program musi podjąć, staje się ogromnym zadaniem.

# Bazy wiedzy

---

- „Efektywny” model algorytmicznej reprezentacji wiedzy wciąż czeka na odkrycie...