12 Datové struktury

Data, se kterými pracují programy (algoritmy) jsou uchovávána v **datových strukturách** a lze je prezentovat **různými datovými typy**.

Pro efektivní práci s daty je nutné zvolit:

- odpovídající datový typ
- vhodnou datovou strukturu.

Datový typ – určuje v programování rozsah a druh hodnot, kterých nabývá datová struktura (např. proměnná). Datový typ je omezen oborem hodnot a zjednodušeně se jedná o to, jaké typy hodnot lze do jednotlivých struktur (proměnných) přiřazovat. Zároveň má každý datový typ definováno (určuje), jaké operace s ním lze a nelze provádět (například že nelze od sebe odečíst dvě proměnné obsahující řetězec znaků). V základním rozdělení existují dva druhy datových typů: **jednoduché** a **složené datové typy**. **Jednoduché datové typy** jsou jasně definovány a zakomponovány přímo v příslušném programovacím jazyce, kde je uživatel využívá. Různé jazyky mohou mít různé datové typy. Základními datovými typy jsou: logická hodnota (pravda / nepravda), celé číslo, reálné číslo, znak a výčet prvků. Existuje také prázdný datový typ, který nenabývá žádných hodnot. **Složené datové typy** vznikají skládáním výše uvedených jednoduchých typů.

DATOVÉ STRUKTURY MOHOU BÝT:

Samostatné hodnoty (**Jednoduché typy**) – charakterizují množiny hodnot, které lze nazvat hodnotami elementárními, a jsou tvořeny jednoduchými datovými typy, jako jsou čísla, písmena, znaky apod.

Strukturované údaje (**Strukturované typy**) – charakterizují množiny struktur (uskupení) a mají charakter záznamů. Záznamy většinou popisují nějaké objekty. Je v nich více položek - **datových členů**, přičemž každý člen popisuje nějaký rys nebo hodnotu daného objektu. Například bude-li záznam popisovat osobu, pak typicky bude obsahovat její rodné číslo, jméno a příjmení, jednotlivé části adresy bydliště (název ulice s popisným číslem, PSČ a název města) atd.

Pro popis algoritmů většinou není důležité, pro jaké datové typy budou prakticky použity, zda pro jednoduché nebo pro strukturované datové typy. Budou-li pracovat s nějakými prvky z nějaké množiny dat. Nicméně aby algoritmy mohly prvky vyhledávat nebo je srovnávat, potřebují, aby u každého prvku byla stanovena hodnota, která ho reprezentuje. U prvků, jež jsou tvořeny **jednoduchými datovými typy**, **je touto hodnotou samotná hodnota prvku**. **U strukturovaného typu je touto hodnotou většinou nějaký jeho člen (nebo několik jeho členů)**, a to takový, že jeho hodnota prvek jednoznačně určuje. Hodnotě, která prvek reprezentuje (určuje), budeme říkat **klíč prvku**.

ZÁKLADNÍ ROZDĚLENÍ DATOVÝCH STRUKTUR:

1. **Základní datové struktury** – vyskytují se téměř ve všech programovacích jazycích. Za běhu programu nemění svůj rozsah. Např. **proměnná**, **pole**, **záznam** a **objekt**.
2. **Odvozené datové struktury** – nazývány jako abstraktní datové struktury. Často implementovány jako objekty. Za běhu programu mohou měnit svůj rozsah. Např. **seznam**, **zásobník, fronta a strom**.

ČLENĚNÍ DLE ZPŮSOBU USPOŘÁDÁNÍ DAT:

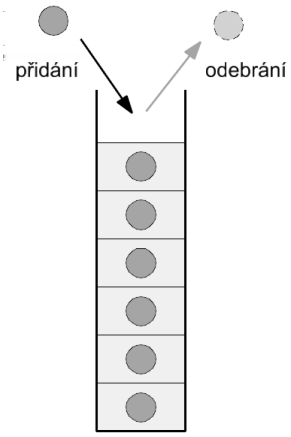
1. **Datová struktura** je považována za **lineární,** pokud datové prvky vytvářejí sekvenci lineárního seznamu. Prvky jsou přiléhající k sobě a ve specifikovaném pořadí. To spotřebovává lineární paměťový prostor, datové prvky jsou vyžadovány ukládat postupně v paměti. Při implementaci lineární datové struktury je předem deklarováno potřebné množství paměti. Nevytváří dobré využití paměti a vede k plýtvání paměti. Datový prvek je navštěvován postupně, kde lze přímo dosáhnout pouze jednoho prvku. Příkladem lineární datové struktury jsou **pole, zásobník, fronta, propojený seznam**, atd.
2. **Nelineární datová struktura** neuspořádá data postupně, ale je uspořádána v tříděném pořadí. V tomto mohou být datové prvky připojeny k více než jednomu prvku vykazujícímu hierarchický vztah, který zahrnuje vztah mezi dítětem, rodičem a prarodičem. V nelineární datové struktuře se průchod datových prvků a vkládání nebo delece neprovádí postupně. Nelineární datová struktura využívá paměť efektivně a nevyžaduje předem deklaraci paměti. Existují dva běžné příklady nelineární datové struktury **strom** a **graf**. Stromová datová struktura organizuje a ukládá datové prvky v hierarchickém vztahu.

KLÍČOVÉ ROZDÍLY MEZI LINEÁRNÍ A NELINEÁRNÍ STRUKTUROU DAT

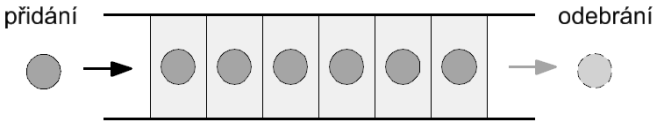
1. V lineární datové struktuře jsou data uspořádána v lineárním pořadí, ve kterém jsou prvky vzájemně propojeny. Proti tomu v nelineární datové struktuře nejsou datové prvky ukládány sekvenčně, ale prvky jsou hierarchicky příbuzné.
2. Přechod dat v lineární datové struktuře je snadný, protože umožňuje, aby všechny datové prvky procházely jedním pohybem, ale současně je přímo dosažitelný pouze jeden prvek. Naopak v nelineární datové struktuře nejsou uzly navštěvovány sekvenčně a nemohou být procházeny v jednom kroku.
3. Datové prvky jsou vedle sebe připojeny v lineární datové struktuře, což znamená, že pouze dva elementy mohou být spojeny se dvěma dalšími prvky, zatímco toto není případ nelineární datové struktury, kde jeden datový prvek může být připojen k mnoha dalším prvkům.
4. Lineární datové struktury jsou snadno implementovány vzhledem k nelineární datové struktuře.
5. Do lineární datové struktury je začleněna jediná úroveň prvků. Naopak nelineární datová struktura zahrnuje více úrovní.
6. Příklady lineární datové struktury jsou pole, fronta, zásobník, propojený seznam, atd. Naproti tomu strom a graf jsou příklady nelineární datové struktury.
7. Paměť je efektivně využívána v nelineární datové struktuře, kde lineární datová struktura má tendenci plýtvat pamětí.

ČLENĚNÍ DLE ZPŮSOBU MANIPULACE S DATY

1. **LIFO** (z anglického „Last In – First Out“) – charakteristický způsob manipulace s daty je, že data uložena jako poslední budou čtena jako první. Příkladem takovéto datové struktury je **zásobník.**



2. **FIFO** (z anglického „First In – First Out“) – charakteristický způsob manipulace s daty je, že data uložena jako první budou čtena jako první. Příkladem takovéto datové struktury je **fronta.**



PROMĚNNÁ (SAMOSTATNÁ HODNOTA)

Proměnná představuje pojmenované místo v paměti počítače. Je v ní uložena hodnota určitého datového typu. Vytvoří se na základě **deklarace**, ve které sdělíme její **jméno** (odborně ***identifikátor)*** a **datový typ**. Počítač (procesor) zachází s proměnnou prostřednictvím její **adresy**.

Datový typ proměnné je dán:

1. **Množinou hodnot**, kterých proměnná může nabývat.
2. **Množinou operací**, které lze s danou proměnnou provádět.
3. **Množstvím paměti** (alokace paměti) potřebné pro její uložení.

DRUHY PROMĚNNÝCH

Ve většině programovacích jazyků se setkáme se třemi základními druhy proměnných, které se liší způsobem **alokace** = **přidělení paměti** (způsobem, jakým je proměnným přidělován paměťový prostor v RAM):

1. Globální (veřejná) proměnná – vytvoří se při spuštění programu a existuje po celou dobu běhu programu. Zaniká s jeho ukončením,
2. Lokální proměnná – je proměnná deklarovaná v procedurách nebo funkcích. Lokální proměnné vznikají v okamžiku volání podprogramu a při ukončení podprogramu zanikají.

3. Dynamická proměnná – vznikají za běhu programu na základě příkazů. Podobně na základě příkazů programu i zanikají.

ZÁVĚR

Ukládání a vyhledávání může být prováděno nad daty uloženými v hlavní paměti nebo v sekundární paměti; podle typu paměti volíme vhodné **datové struktury** a **algoritmy**. Při vývoji softwaru závisí složitost implementace a rychlost práce výsledného programu na správném výběru **datových** struktur.

Kritéria pro návrh datových struktur:

- rychlost čtení (včetně nalezení dat),
- rychlost zápisu (operace vložení, mazání, aktualizace),
- paměťová náročnost,
- náročnost implementace (čím komplikovanější algoritmus, tím větší pravděpodobnost chyby).

Existuje celá řada datových struktur. Mnohé „klasické“ datové struktury jsou obsaženy buď ve standardních knihovnách programovacích jazyků, nebo vestavěny přímo v programovacích jazycích.