Struktury dynamiczne

O strukturach dynamicznych mówimy wtedy, gdy wielkość i stopień złożoności struktur danych zmienia się podczas działania programu. Do najważniejszych struktur dynamicznych zaliczamy: *stos, kolejkę, listę i drzewo*.

*Przypomnienie:* Do tej poznaliśmy możliwość dynamicznego przydziału pamięci dla zmiennych różnych typów tzw. *zmiennych dynamicznych*. Przykładem mogą być *tablice dynamiczne*, o których tworzeniu i rozmiarze możemy decydować w trakcie realizacji programu.

Do dynamicznego przydziału i zwalniania pamięci stosuje się operatory new i delete.

Operator **new** służy do przydzielania pamięci dla zmiennej dynamicznej.

Operator **delete** wykorzystywany jest do zwalniania pamięci (usuwa zmienna dynamiczną).

*Przykład* Utworzenie zmiennej dynamicznej typu int.

int \*wsk; //deklaracja wskaźnika, gdyż zmienne dynamiczne nie mają nazw

wsk = new int; //przydział pamięci na zmienną – utworzenie zmiennej dynamicznej

\*wsk = 10;

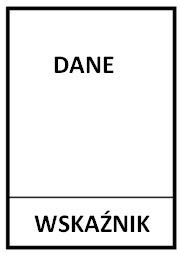
cout<<\*wsk; //wykorzystanie zmiennej dynamicznej

delete wsk; //usunięcie zmiennej dynamicznej

**Zmienne dynamiczne są jedynie składnikami dynamicznych struktur danych.**

Struktury dynamiczne stosujemy w momencie, gdy pisząc kod programu nie wiemy jeszcze, ile danych będziemy zmuszeni przechowywać lub jakie dokładnie zależności pomiędzy tymi danymi będą zachodziły (np. reprezentując drzewo genealogiczne nie wiemy w czasie pisania programu, ile dana osoba ma rodzeństwa).

*Budowa:* Nieważne o jakiej strukturze dynamicznej mowa, można przyjąć, z każda składa się z *elementów*. Stos i kolejka mają elementy dwuczęściowe, drzewo binarne – trzyczęściowe, lista – dwu- lub trzyczęściowe. Dalej pokazane będą elementy dwuczęściowe.



W C++ do realizacji takiego elementu stosuje się struktury. Część do przechowywania danych może być dowolnego typu, w tym również strukturą.

struct element

{

string dane;

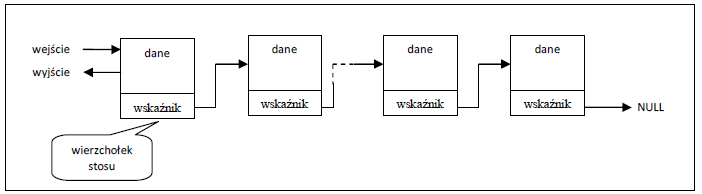
element \*wsk;

};

**DANE**

Elementy więcej niż dwuczęściowe mają po prostu więcej wskaźników.

**STOS** <http://miroslawzelent.pl/kurs-c++/struktury-danych-stos-kolejka-lista-drzewo-binarne/>

Stos (Last In First Out, LIFO) to dynamiczna struktura danych, w której elementy są liniowo uporządkowane oraz dostęp mamy tylko do tego elementu, który został dołączony jako ostatni tzw. wierzchołka.

Na rysunku przedstawiono budowę stosu. Każdy element stosu będący strukturą składa się z **danych** oraz pola będącego **wskaźnikiem**, który zawiera adres kolejnego elementu. Ostatni element stosu zawiera wskaźnik pusty równy NULL.

Miejsce, do którego mamy dostęp, to wierzchołek stosu. Wierzchołek stosu to **jedyne** miejsce w stosie, na którym można wykonywać operacje: dołączać i usuwać elementy stosu.

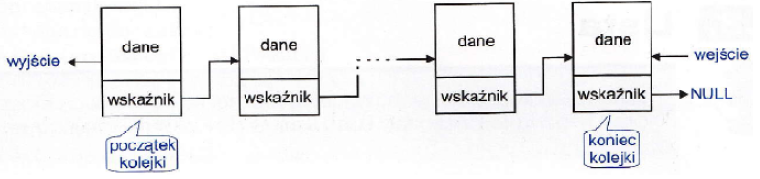
***Ćwiczenie 1*** Wykonać program do przechowywania tekstów na stosie. Program ma posiadać funkcje:

* push() – odkłada tekst na stos czyli dodaje tekst na wierzchołek;
* pop() – usuwa tekst ze stosu czyli zdejmuje go z wierzchołka;
* size() – podaje ile elementów (czyli tekstów) jest na stosie;
* empty() – sprawdza czy stos jest pusty?

Program powinien także pokazywać na ekranie aktualny stan stosu (jego zawartość).

**KOLEJKA** *17m48s*

Kolejka (First In First Out, FIFO) to dynamiczna struktura danych, w której element są liniowo uporządkowane. Dołączać je można **tylko na końcu**, a usuwać **wyłącznie z początku** kolejki.



Każdy element kolejki będący strukturą składa się z danych oraz pola będącego wskaźnikiem, który zawiera adres następnego składnika. Każdy element kolejki (oprócz ostatniego) zawiera wskaźnik wskazujący na następny składnik. Ostatni element kolejki, znajdujący się na końcu, zawiera wskaźnik pusty równy NULL.

***Ćwiczenie 2*** Wykonać program do przechowywania tekstów w kolejce. Program ma posiadać funkcje:

* dodaj() – dodaje tekst do kolejki czyli na jej końcu;
* usun() – usuwa tekst z kolejki czyli z jej początku;
* size() – podaje ile elementów (czyli tekstów) jest w kolejce;
* empty() – sprawdza czy kolejka jest pusta?

Program powinien także pokazywać na ekranie aktualny stan kolejki (jej zawartość).

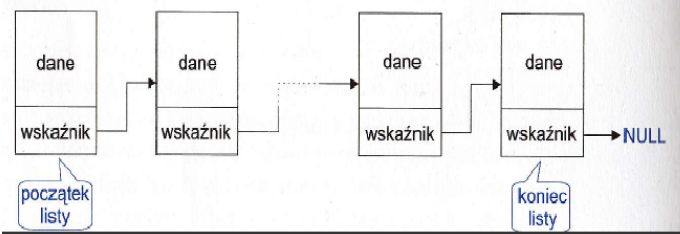
**LISTY** *24m50s*

Lista to struktura danych, w której elementy są liniowo uporządkowane oraz można je dołączyć i usuwać w dowolnym miejscu listy.

Ze względu na zależność między elementami listy wyróżniamy:

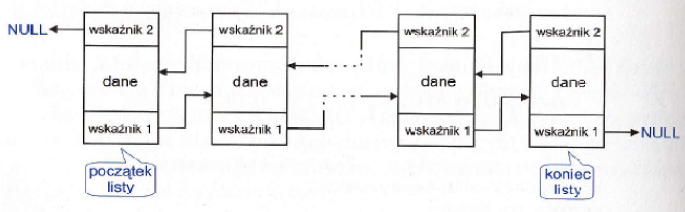
* listę jednokierunkową,
* listę dwukierunkową,
* listę cykliczną

**Lista jednokierunkowa**



Każdy element listy zawiera dane oraz wskaźnik na następny element. Ostatni składnik listy zawiera wskaźnik pusty równy NULL. Nie ma określonego miejsca w którym możemy dodawać lub usuwać składnik do listy.

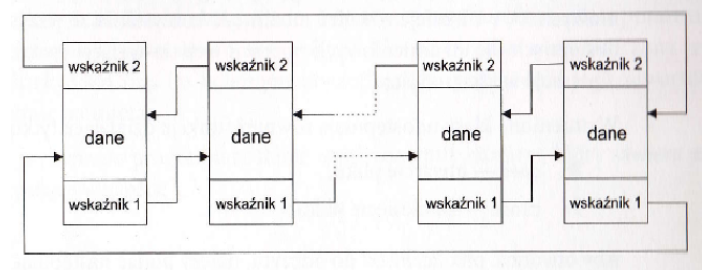
**Lista dwukierunkowa**



Listę dwukierunkową charakteryzuje się tym, że każdy jej element (poza skrajnymi) ma określony poprzednik i następnik. Stąd każdy element listy dwukierunkowej musi zawierać dwa wskaźniki.

**Lista cykliczna**

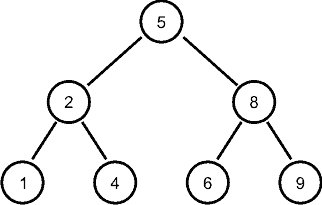
Lista ta nie zawiera elementów, które są początkiem lub końcem listy. Każdy element listy ma poprzednik i następnik.



***Ćwiczenie 3*** Wykonać program do przechowywania tekstów na liście jednokierunkowej. Program ma posiadać funkcje:

* dodaj\_na\_poczatku() – dodaje element na początku listy;
* dodaj\_na\_koncu() – dodaje element na końcu listy;
* usun\_z\_poczatku() – usuwa element z początku listy;
* usun\_z\_konca() – usuwa element z końca listy;
* sortuj() – sortuje elementy na liście np. rosnąco;
* zapisz\_do\_pliku() – zapisuje dane z listy do pliku;
* odczytaj\_z\_pliku – odczytuje dane z pliku i tworzy z nich listę;
* size() – podaje ile elementów jest na liście;
* empty() – sprawdza, czy lista jest pusta?

Program powinien także pokazywać na ekranie aktualny stan listy (jej zawartość). Pomoc i wskazówki: <http://eduinf.waw.pl/inf/alg/001_search/0086.php#P7>

**DRZEWO** *37m50s*

**Drzewo** jest hierarchiczną strukturą danych zbudowaną z elementów, które nazywamy **węzłami** (ang. node). Dane przechowuje się w węzłach drzewa. Węzły są ze sobą powiązane w sposób hierarchiczny za pomocą **krawędzi**, które zwykle przedstawia się za pomocą strzałki określającej hierarchię. Pierwszy węzeł drzewa nazywa się **korzeniem** (ang. root). Od niego "wyrastają" pozostałe węzły, które będziemy nazywać **synami**.

Dalej ograniczymy się do drzewa binarnego czyli takiego, w którym węzły mogą posiadać **co najwyżej dwóch** synów (patrz rysunek). Więcej tutaj: <http://eduinf.waw.pl/inf/alg/001_search/0107.php>

***Ćwiczenie 4*** Wypróbuj działania programu z filmu i przeanalizuj jego kod.