Temat: Dynamiczne struktury danych

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

- ✓ stos
- ✓ kolejka
- ✓ lista
- ✓ drzewo

Istnieje ponadto możliwość dynamicznej alokacji pamięci dla zmiennych różnych typów. 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 i ma składnię:

identyfikator = new typ_obiektu;

Wartością wyrażenia jest tutaj wskaźnik do utworzonego obiektu.

Operator **delete** wykorzystywany jest do zwalniania pamięci i ma następującą składnię:

delete identyfikator;

Operator delete usuwa obiekt, który jest wskazany przez identyfikator.

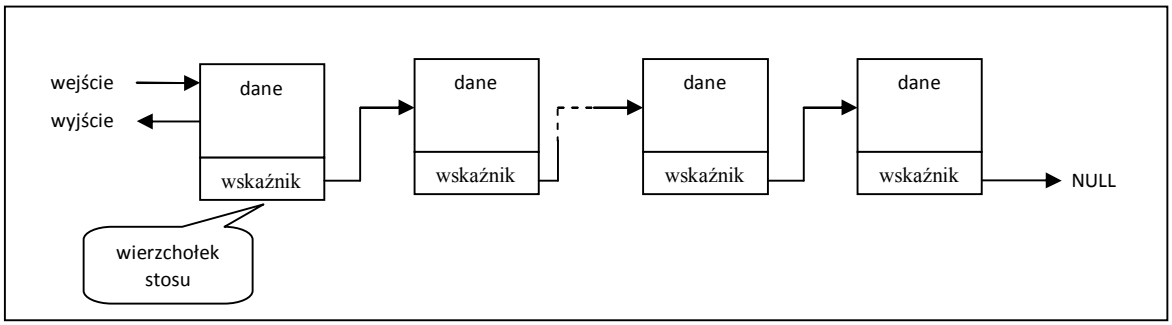
Przykład.

Dynamiczna alokacja pamięci dla zmiennej typu int:

```
int *wsk;
wsk = new int;
delete wsk;
```

Temat: Stos – ostatni wchodzi, pierwszy wychodzi (LIFO).

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.



Na rysunku przedstawiono budowę stosu. Każdy element stosu będący strukturą składa się z **danych** oraz **pola będącego wskaźnikiem**, który zawiera adres poprzedniego składnika. 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.

Definicja struktury reprezentującej składnik stosu:

```
struct  element
      { int wartosc;
       element *poprzedni;
      };
```

Funkcja realizująca dodawanie do stosu:

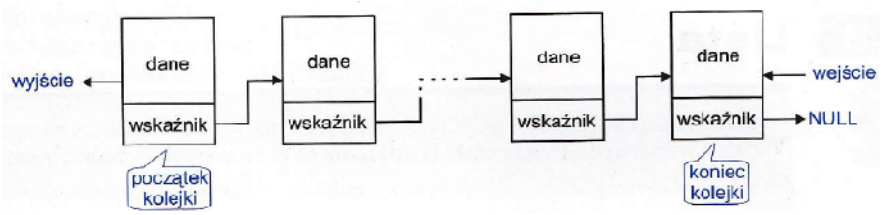
```
element *dodaj (int liczba, element *wierzcholek)
{
   element *wsk;
  wsk = new element;
  wsk ->wartosc=liczba;
  wsk->poprzedni=wierzcholek;
  return wsk;
}
```

Funkcja usuwa element ze stosu:

```
element *usun (int *liczba, element *wierzholek)
{
  element *wsk;
  *liczba=wierzcholek->wartosc;
  wsk=wierzholek->poprzedni;
  delete wierzcholek;
  return wsk;
}
```

Temat: Kolejki – pierwszy wchodzi pierwszy wychodzi (FIFO).

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.



Rysunek przedstawia kolejki. Każdy element kolejki będący strukturą składa się z danych oraz pola będącego wskaźnikiem, który zawiera adres następnego składnika. Na początku kolejki znajduje się element, który zawiera wskaźnik wskazujący na następny składnik. Ostatni element kolejki, znajdujący się na końcu, zawiera wskaźnik pusty równy NULL. Trzeba pamiętać, że elementy można dołączać tylko na końcu kolejki a usuwać na początku.

Definicja struktury reprezentującej składnik kolejki:

```
struct  element
      { int wartosc;
       element *nastepny;
      };
```

Funkcja realizująca dodawanie do kolejki:

```
element *dodaj (int liczba, element *koniec)
{
   element *wsk;
  wsk = new element;
  wsk ->wartosc=liczba;
  wsk->nastepny=NULL;
  return wsk;
}
```

Funkcja usuwa element ze stosu:

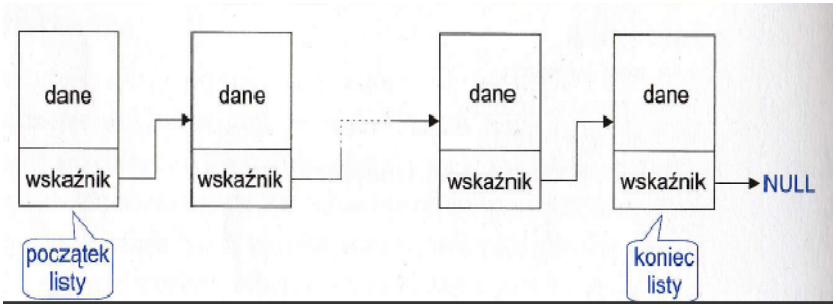
```
element *usun (int *liczba, element *poczatek)
{
  if (poczatek)
   {
     element *wsk;
    *liczba=poczatek->wartosc;
      wsk= poczatek ->nastepny;
     delete pocztek;
  return wsk;
}
```

Temat: Lista jednokierunkowa, dwukierunkowa i cykliczna.

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:

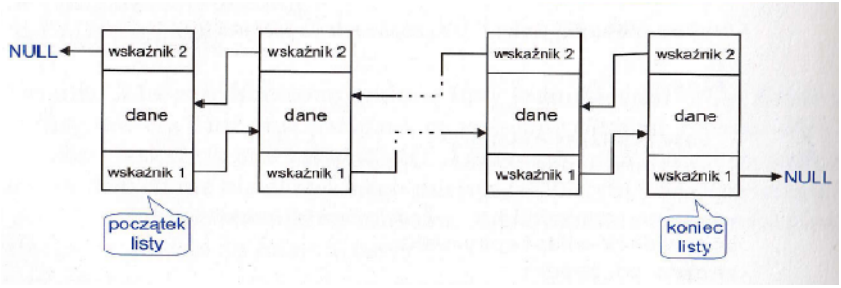
- ✓ listę jednokierunkową,
- ✓ listę dwukierunkową,
- ✓ listę cykliczną

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: wskaźnik1 – adres poprzednika (na początku listy jest on równy NULL), wskaźnik2 – adres następnika (na końcu listy wynosi NULL).

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