**Wskaźnik** (ang. *pointer*) jest adresem pojedynczej komórki pamięci operacyjnej.
(Jest to więc w istocie liczba, interpretowana jako unikalny indeks danego obszaru w pamięci.)
W języku C++ każdy wskaźnik musi mieć dodatkowo określony typ, na jaki wskazuje, np.:
int * ptr;

- **operator pobierania adresu, oznaczanego znakiem & (ampersandem).**

```
// zadeklarowanie zmiennej oraz odpowiedniego wskaźnika
unsigned uZmienna;
unsigned* puWskaznik;
 // pobranie adresu zmiennej i zapisanie go we wskaźniku
 puWskaznik = &uZmienna;
```

- **operator dereferencji wskaźnika, symbolem którego jest***

```
// zapisanie wartości w komórce pamięci, na którą pokazuje wskaźnik
*puWskaznik = 123;
// odczytanie i wyświetlenie tej wartości
std::cout << "Wartosc zmiennej uZmienna: " << *puWskaznik;
```

- **Operator wyłuskania „->" służy do wybierania składników obiektu, na który wskazuje wskaźnik.**

Pod pojęciem 'obiektu' kryje się tu zarówno instancja klasy, jak i typu strukturalnego.

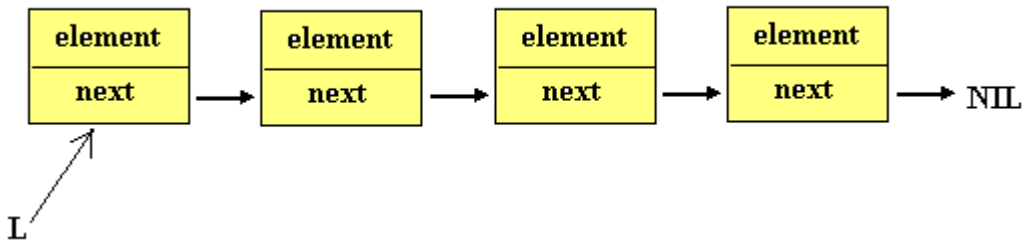
```
typedef  struct celltype
{int element;
 celltype * next;}
celltype * l;//l jest zmienną wskaźnikową – zawiera adres komórki typu celltype

l->element;//wybieramy pole element z obiektu na kóry wskazuje l
l->next; //wybieramy pole next z obiektu na kóry wskazuje l
```

**Lista – ciąg 0 lub więcej elementowy, złożony z elementów tego samego typu: a1 ,a2,...,an n>=0;**
**Operacje na liście:**

- **INSERT(x,p,L) – wstaw *x* na pozycję *p* na liście *L***
- **LOCATE(x,L) zwraca pozycję pierwszego wystąpienia elementu *x* w liście *L*.**
- **RETRIEVE(p,L) – zwraca element występujący w *L* na pozycji *p*.**
- **DELETE(p,L) – usuwa element na pozycji *p* z listy.**
- **NEXT(p,L) – zwraca pozycję następną w stosunku do *p* w *L*.**
- **PREVIOUS(p,L)- zwraca pozycję następną w stosunku do *p* w L**
- **MAKENULL(L).Czyni listę pustą i zwraca pozycję END(L).**
- **FIRST(L) – zwraca pozycję pierwszego elementu w L.**
- **PRINT_LIST(L) – wypisuje elementy w kolejności występowania**

**Lista wskaźnikowa – ciąg zmiennych dynamicznych powiązanych wiązaniami. Jedna zmienna wskazuje na drugą dzięki polu typu wskaźnikowego, który jest jej składnikiem.**




**Zmienne dynamiczne:**
rezerwowanie i zwalnianie pamięci w trakcie działania programu
Operator new – przydziela pamięć dla obiektów.
Operator delete – zwalnia pamięć przydzieloną za pomocą operatora new (wynik działania jest typu void).

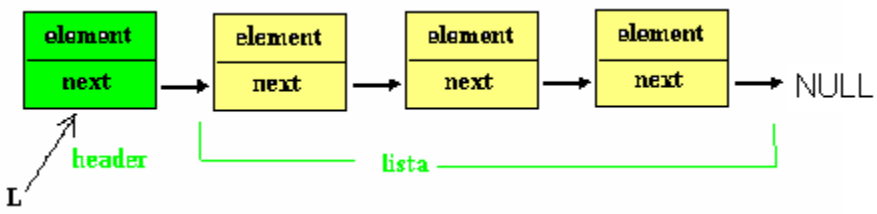
```
int *pi;
pi = new int;
delete pi;
```

**Przykładowa definicja typów:**

```
typedef int  elementtype;
struct celltype {
elementtype element;
celltype * next;
};

typedef celltype  * position;
```

**Lista z głową:**
Pozycją elementu i-tego jest wskaźnik do elementu i-1.
Pozycją elementu pierwszego jest wskaźnik do komórki header.
ENDL jest wskaźnikiem do ostatniej komórki listy.
Lista w tej implementacji jest wskaźnikiem do głowy (header)

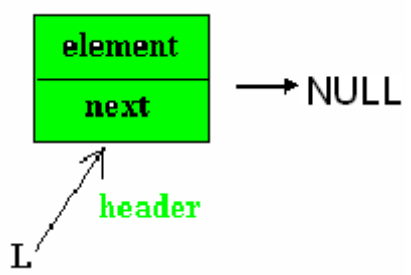


```
typedef int  elementtype;
struct celltype {
elementtype element;
celltype * next;
};

typedef celltype  * position;

class Lista
{
protected :
position l; // wskaźnik do głowy listy
public:
    Lista();    //konstruktor
    ~Lista();  //destruktor
    void Insert(elementtype x, position p);
    void Delete(position p);
    elementtype Retrieve(position p);
    position Locate(elementtype x);
    position First();
    position Next(position p);
    position Previous(position p);
    position END();
};
```

**Lista pusta:**



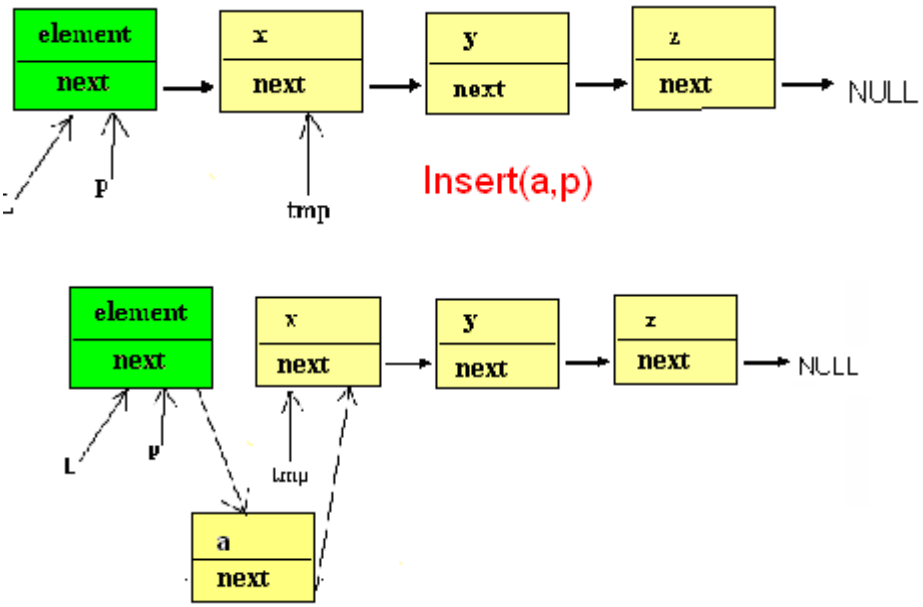

**Zad. Zaimplementuj operacje na liście (lista wskaźnikowa z głową)**

**Uwagi pomocnicze:**

- pozycja elementu- wskaźnik do elementu poprzedniego
- pozycja elementu pierwszego - wskaźnik do głowy listy
- END() - zwraca pozycję za ostatnim elementem (czyli wskaźnik do ostatniego eleemntu)
- Next(p) -zwraca wskaźnik do  nastepnego elementu po p (czyli p->next):

- Previous(p) - zwraca pozycję poprzedniego elementu w stosunku do p
  (przegląda całą listę, aby znaleźć takie tmp, że tmp->next = =p)

- Insert(x,p) - wstawia x na pozycję p-czyli wstawiamy nową komórkę za komórką
  wskazywaną przez p




- Delete(p) - usuwa z pozycji p, czyli usuwa  komórkę wskazywaną przez p->next:

- Locate(x ) - zwraca pozycję elementu x w liście, czyli taką pozycję p, że
  p->next->element= =x
  (pętla po wszystkich elementach listy, aż znajdziemy p takie, że p->next->element= =x )

- Retrieve(p) - zwraca element znajdujący się w liście na pozycji p(w komórce p->next),
  jeśli p->next!=NULL