// PROGRAM 4_1 - Przyklady dynamicznego tworzenia
// i usuwania tablicy dwuwymiarowej int [2][3]

```
#include <alloc.h>

void main(void)
{
```

//---------- Przyklad A ----------
//zwykla tablica dwuwymiarowa (dla porównania)

```
int tab_A [ 2 ] [ 3 ];

// przykladowe operacje
tab_A[0][0] = 200;
tab_A[0][1] = 201;
tab_A[1][0] = 210;
tab_A[1][1] = tab_A[1][1];

// nie trzeba oprogramowywac operacji usuwania tablicy A
// bo jej tworzenie i usuwanie jest zrealizowane przez kompilator !
```

//---------- Przyklad B ----------
// wskaznik na dwuwymiarowa dynamiczna tablice liczb calkowitych
// Uwaga: taka reprezentacja jest bardzo niewygodna do zapisywania
// bo wszedzie trzeba dopisywac operator „gwiazdki” lub [0]

```
int (*tab_B) [ 2 ] [ 3 ];

tab_B = ( int(*)[2][3] )malloc( 2*3*sizeof(int) );
// ##  tab_B = new int[1][2][3];    << to samo, za pomoca operatora „new”

// przykladowe operacje na takiej tablicy B
(*tab_B)[0][0] = 200;                    // tab_B[0][0][0] = 200;
(*tab_B)[0][1] = 201;                    // tab_B[0][0][1] = 201;
(*tab_B)[1][0] = 210;                    // tab_B[0][1][0] = 210;
(*tab_B)[1][1] = (*tab_B)[1][1];

// zwolnienie obszaru zajmowanego przez tablice B
free( tab_B );
// ##  delete tab_B;       << to samo, za pomoca operatora „delete”
```

//---------- Przyklad C ----------
//wskaznik na pierwsza 3-elementowa tablice (pierwsza z dwóch)

int (*tab_C) [3];

```
tab_C = ( int(*)[3] )malloc( 2*3*sizeof(int) );
// ##  tab_C = new int [2][3];

// przykladowe operacje na tablicy C
tab_C[0][0] = 200;
tab_C[0][1] = 201;
tab_C[1][0] = 210;
tab_C[1][1] = tab_C[1][1];

// zwolnienie obszaru zajmowanego przez tablice C
free( tab_C );
// ##  delete tab_C;
```

//---------- Przyklad D ----------
//zwykla dwuelementowa tablica wskazników na dynamiczne tablice liczb

int *tab_D [2];

```
tab_D[0] = (int*)malloc( 3*sizeof(int) );  // pierwszy wiersz
tab_D[1] = (int*)malloc( 3*sizeof(int) );  // drugi wiersz

// ##  tab_D[0] = new int [3];  // pierwszy wiersz
// ##  tab_D[1] = new int [3];  // drugi wiersz

// przykladowe operacje na tablicy D
tab_D[0][0] = 200;
tab_D[0][1] = 201;
tab_D[1][0] = 210;
tab_D[1][1] = tab_D[1][1];

// zwolnienie obszaru zajmowanego przez tablice D
free( tab_D[0] );
free( tab_D[1] );

// ##  delete tab_D[0];
// ##  delete tab_D[1];
```

//---------- Przyklad E ----------
// dynamiczna dwuelementowa tablica wskazników na dynamiczne tablice
// 3 liczb calkowitych

int **tab_E;

// Dwuetapowe tworzenie tablicy -> najpierw tablica z adresami wierszy
tab_E = (int**)malloc(2*sizeof(int*)); *// tablica dwóch wskazników*

// potem dwie tablice zawierajace wiersze
tab_E[0] = (int*)malloc(3*sizeof(int)); *// pierwszy wiersz*
tab_E[1] = (int*)malloc(3*sizeof(int)); *// drugi wiersz*

// ## tab_E = new int [2]; // tablica dwóch wskazników*
// ## tab_E[0] = new int [3]; // pierwszy wiersz
// ## tab_E[1] = new int [3]; // drugi wiersz

// przykladowe operacje na tablicy E
tab_E[0][0] = 200;
tab_E[0][1] = 201;
tab_E[1][0] = 210;
tab_E[1][1] = tab_E[1][1];

// zwolnienie obszaru zajmowanego przez tablice E
free(tab_E[0]); *// zwolnienie pierwszego wiersza*
free(tab_E[1]); *// zwolnienie drugiego wiersza*
free(tab_E); *// dopiero na koniec, zwolnienie tablicy wskazników*

// ## delete tab_E[0];
// ## delete tab_E[1];
// ## delete tab_E ;

} *// main – program 4_1*

```
//  PROGRAM 4_2
//  Przyklad oprogramowania dynamicznej listy jednokierunkowej
//  ( najprostsza wersja bez wyróznienia funkcji – interfejsu )

#include <iostream.h>
#include <conio.h>      // getch()
#include <ctype.h>      // toupper()

struct TDane
{
 char tekst[30];

 int  liczba;
 char znak;
 // . . . inne dane
};

struct TElementListy
{
 TDane dane;
 TElementListy* nastepny;
};

void main()
{
 TElementListy* pierwszy=NULL;

 //-------- FAZA WCZYTYWANIA DANYCH I TWORZENIA LISTY ----------
 int klawisz;
 while(1)
  {
   cout<<"\n\nCzy chcesz dodac nowy element do listy (t/n) : ";
   klawisz = toupper( getch() );
   if( klawisz=='N' )
     break;

   // wczytywanie danych kolejnego elementu
   TDane nowe_dane;
   cout<<"\n\nWprowadznie danych nowego elementu:\n";
   cout<<"Podaj tekst: ";      cin>>nowe_dane.tekst;
   cout<<"Podaj liczbe: ";     cin>>nowe_dane.liczba;
   cout<<"Podaj znak: ";       cin>>nowe_dane.znak;

   // przydzielenie pamieci na nowy element listy
   TElementListy* nowy_element;
   nowy_element = new TElementListy;
```

```
   if( nowy_element==NULL )
     {
      cout<<"\n\nBlad przydzialu pamieci. Koniec wprowadzania.\n";
      break;
     }

   // skopiowanie danych i dolaczenie nowego elementu do listy
   nowy_element->dane = nowe_dane;
   nowy_element->nastepny = pierwszy;
   pierwszy = nowy_element;
  }

//----------  FAZA WYSWIETLANIA ZAWARTOSCI LISTY  ----------------
if( pierwszy==NULL )
  cout<<"\nLista jest pusta\n";
else
  {
   cout<<"\nLista zawiera nastepujace elementy:";
   TElementListy* aktualny;
   int licznik=0;
   for( aktualny=pierwszy; aktualny!=NULL; aktualny=aktualny->nastepny )
     {
      cout<<"\n\nElement numer: "<< ++licznik;
      cout<<"\n       tekst: "<< aktualny->dane.tekst;
      cout<<"\n      liczba: "<< aktualny->dane.liczba;
      cout<<"\n        znak: "<< aktualny->dane.znak;
     }
  }
cout<<"\nNacisnij dowolny klawisz\n";
getch();

//----------  FAZA USUWANIA LISTY - ZWALNIANIA PAMIECI  ----------------
TElementListy* aktualny;
TElementListy* do_usuniecia;
aktualny=pierwszy;
while( aktualny )
  {
   do_usuniecia = aktualny;
   aktualny = aktualny->nastepny;
   delete do_usuniecia;
  }

cout<<"\nLista zostala usunieta.";
cout<<"\nNacisnij dowolny klawisz aby zakonczyc";
getch();
}
```

```
//  PROGRAM 4_3
//  Przyklad oprogramowania dynamicznej listy jednokierunkowej
//  ( druga wersja - z wyróznieniem funkcji operujacych na
//  strukturze TDane oraz na liscie )

#include <iostream.h>
#include <conio.h>      // getch()
#include <ctype.h>      // toupper()

// Nowy typ "TBoolean" modelujacy wartosci logiczne: prawda / falsz
typedef char TBoolean;
#define  True   0xFF
#define  False  0x00


//  Struktura TDane i funkcje Wczytaj(), Wyswietl()
//  modelujace ”dane” przechowywane w elementach listy

struct TDane
{
 char tekst[30];
 int  liczba;
 char znak;
};


void Wczytaj(TDane &dane)
{
 cout<<"\n\nWprowadznie danych nowego elementu:\n";
 cout<<"Podaj tekst: ";   cin>>dane.tekst;
 cout<<"Podaj liczbe: ";  cin>>dane.liczba;
 cout<<"Podaj znak: ";    cin>>dane.znak;
}


void Wyswietl(const TDane &dane)
{
 cout<<"\n       tekst: " << dane.tekst;
 cout<<"\n      liczba: " << dane.liczba;
 cout<<"\n        znak: " << dane.znak;
}
```

```
// Struktura TElementListy
// i funkcje Dopisz(), Usun() modelujace operacje na "elementach" listy

struct TElementListy
{
 TDane dane;
 TElementListy* nastepny;
};


TBoolean Dopisz(TElementListy* &pierwszy, const TDane& dane)
{
 TElementListy* nowy_element;
 nowy_element = new TElementListy;
 if( nowy_element==NULL )
    return False; //blad przydzialu pamieci

 // skopiowanie danych i dolaczenie nowego elementu do listy
 nowy_element->dane = dane;
 nowy_element->nastepny = pierwszy;
 pierwszy = nowy_element;
 return True;
}


TBoolean UsunElement(TElementListy* &pierwszy)
{
 if( !pierwszy )
   return False;
 else
   {
    TElementListy* do_usuniecia;
    do_usuniecia = pierwszy;
    pierwszy = pierwszy->nastepny;
    delete do_usuniecia;
    return True;
   }
}
```

```
void main()   //---------- Program glowny ----------
{
 TElementListy* pierwszy=NULL;
 // -------- FAZA WCZYTYWANIA DANYCH I TWORZENIA LISTY ----------
 int klawisz;
 while(1)
  {
   cout<<"\n\nCzy chcesz dodac nowy element do listy (t/n) : ";
   klawisz = toupper( getch() );
   if( klawisz=='N' )
     break;
   // wczytywanie danych kolejnego elementu
   TDane nowe_dane;
   Wczytaj( nowe_dane );
   // przydzielenie pamieci, skopiowanie i dodanie nowego elementu do listy
   if( ! Dopisz( pierwszy, nowe_dane ) )
     {
      cout<<"\n\nBlad przydzialu pamieci. Koniec wprowadzania.\n";
      break;
     }
  }//while(1)
 //----------  FAZA WYSWIETLANIA ZAWARTOSCI LISTY  ----------
 if( pierwszy==NULL )
   cout<<"\nLista jest pusta\n";
 else
  {
   cout<<"\nLista zawiera nastepujace elementy:";
   TElementListy* aktualny;
   int licznik=0;
   for( aktualny=pierwszy; aktualny!=NULL; aktualny=aktualny->nastepny)
     {
      cout<<"\n\nElement numer: "<< ++licznik;
      Wyswietl( aktualny->dane );
     }
  }
 cout<<"\nNacisnij dowolny klawisz\n";
 getch();
 //----------  FAZA USUWANIA LISTY - ZWALNIANIA PAMIECI  ----------
 while( pierwszy )
   UsunElement( pierwszy );
 cout<<"\nLista zostala usunieta.";
 cout<<"\nNacisnij dowolny klawisz aby zakonczyc";
 getch();
}
```

```
//   PROGRAM 4_4
//   Przyklad „obiektowego” podejscia do oprogramowania
//   pojecia  TDana  (struktura modelujaca dane)
//   oraz pojecia TLista (struktura modelujaca liste danych)

#include <iostream.h>
#include <conio.h>      // getch()
#include <ctype.h>      // toupper()

// Nowy typ "TBoolean" modelujacy wartosci logiczne: prawda / falsz
typedef char TBoolean;
#define True  0xFF
#define False 0x00



// Struktura TDane z metodami Wczytaj(), Wyswietl() modelujaca "dane"

struct TDane
{
 // dane
      char   tekst[30];
      int      liczba;
      char  znak;
 // metody - podstawowe funkcje operujace na strukturze TDane
      void Wczytaj(void);
      void Wyswietl(void);
};


void TDane::Wczytaj(void)
{
 cout<<"\n\nWprowadznie danych nowego elementu:\n";
 cout<<"Podaj tekst: ";   cin>>tekst;
 cout<<"Podaj liczbe: ";  cin>>liczba;
 cout<<"Podaj znak: ";    cin>>znak;
}


void TDane::Wyswietl(void)
{
 cout<<"\n       tekst: " << tekst;
 cout<<"\n      liczba: " << liczba;
 cout<<"\n        znak: " << znak;
}
```

```
// Pomocnicza struktura TElementListy

struct TElementListy
{
 TDane          dane;
 TElementListy* nastepny;
};

//=======================================================
//    Struktura „TLista”  z metodami Inicjalizuj(), Dopisz(), Wyswietl(),
//    UsunOstatni(), UsunWszystkieElementy(),
//     modelujaca jednokierukowa liste elementów TElementListy
//=======================================================

struct TLista
{
 // dane listy
      TElementListy* pierwszy;
 // metody - podstawowe operacje na liscie
      void        Inicjalizuj(void);
      TBoolean    Dopisz(const TDane& dane);
      TBoolean    UsunElement(void);
      void        UsunWszystkieElementy(void);
      void        Wczytaj(void);
      void        Wyswietl(void);
};

// implementacje w/w metod dla struktury TLista

void TLista::Inicjalizuj(void)
{
 pierwszy=NULL;
} // TLista::Inicjalizuj

TBoolean TLista::Dopisz(const TDane& dane)
{
 TElementListy* nowy_element;
 nowy_element = new TElementListy;
 if( nowy_element==NULL )          // blad przydzialu pamieci
    return False;
 // skopiowanie danych i dolaczenie nowego elementu do listy
 nowy_element->dane = dane;
 nowy_element->nastepny = pierwszy;
 pierwszy = nowy_element;
 return True;
} // TLista::Dopisz
```

```
TBoolean TLista::UsunOstatni(void)
{
 if( !pierwszy )
   return False;
 else
   {
    TElementListy* do_usuniecia;
    do_usuniecia = pierwszy;
    pierwszy = pierwszy->nastepny;
    delete do_usuniecia;
    return True;
   }
} // TLista::UsunOstatni


void TLista::UsunWszystkieElementy(void)
{
 while( pierwszy )
   UsunOstatni();
} // TLista::UsunWszystkieElementy


void TLista::Wczytaj(void)
{
 int klawisz;
 while(1)
   {
    cout<<"\n\nCzy chcesz dodac nowy element do listy (t/n) : ";
    klawisz = toupper( getch() );
    if( klawisz=='N' )
      break;

    // wczytywanie danych kolejnego elementu
    TDane nowe_dane;
    nowe_dane.Wczytaj();

    // przydzielenie pamieci, skopiowanie i dodanie nowego elementu do listy
    if( ! Dopisz( nowe_dane ) )
      {
       cout<<"\n\nBlad przydzialu pamieci. Koniec wprowadzania.\n";
       break;
      }
   } // while(1)
} // TLista::Wczytaj
```

```
void TLista::Wyswietl(void)
{
 if( pierwszy==NULL )
   cout<<"\nLista jest pusta\n";
 else
   {
    cout<<"\nLista zawiera nastepujace elementy:";
    TElementListy* aktualny;
    int licznik=0;
    for( aktualny=pierwszy; aktualny!=NULL; aktualny=aktualny->nastepny)
      {
       cout<<"\n\nElement numer: "<< ++licznik;
       aktualny->dane.Wyswietl();
      }
   }
 cout<<"\nNacisnij dowolny klawisz\n";
 getch();
} // TLista::Wyswietl


//------------------------------------  Program glowny ---------------------------------------

void main()
{
 TLista lista;

 lista.Inicjalizuj();

 //-------- FAZA WCZYTYWANIA DANYCH I TWORZENIA LISTY ----------
 lista.Wczytaj();

 //----------  FAZA WYSWIETLANIA ZAWARTOSCI LISY  ---------------
 lista.Wyswietl();

 //----------  FAZA USUWANIA LISTY - ZWALNIANIA PAMIECI  ---------------
 lista.UsunWszystkieElementy();

 cout<<"\nLista zostala usunieta.";
 cout<<"\nNacisnij dowolny klawisz aby zakonczyc";
 getch();
}
```