```
/*   PRZYKŁAD 1.
     WYKORZYSTANIA TYPU STRUKTURALNEGO DO
     PRZEKAZYWANIA ZŁOŻONYCH PARAMETRÓW DO/Z FUNKCJI    */

#include <iostream.h>
#include <math.h>

//definicja zbioru stałych typu "int"
enum T_ROZWIAZANIA {
                    TO_NIE_TROJMIAN=-1,
                    BRAK_ROZWIAZAN,       // = TO_NIE_TROJMIAN + 1
                    JEDNO_ROZWIAZANIE,    // = 1
                    DWA_ROZWIAZANIA };    // = 2


//klasyczna definicja struktury (jedyna możliwa dla "C")
struct  T_ROZWIAZANIE_TROJMIANU
  {
   T_ROZWIAZANIA ilosc_rozwiazan;
   double        x1,x2;
  };


//definicja struktury z poleceniem "typedef"   (C++)
typedef struct
  {
   double a,b,c;
  } T_TROJKA_LICZB;



T_TROJKA_LICZB Wczytaj3liczby(char *napis)
{//  wyświetlenie tekstu zachęty dla użytkownika
 //  i wczytanie z klawiatury trzech liczb rzeczywistych
 T_TROJKA_LICZB trojka;
 cout << napis << endl;
 cout << "A = ";   cin >> trojka.a;
 cout << "B = ";   cin >> trojka.b;
 cout << "C = ";   cin >> trojka.c;
 return trojka;
}
```

```
T_ROZWIAZANIE_TROJMIANU Trojmian( T_TROJKA_LICZB t )
{
 T_ROZWIAZANIE_TROJMIANU rozw={TO_NIE_TROJMIAN,0,0};
 double delta;
 if( t.a==0 )
   return rozw;
 delta = pow(t.b,2) + 4 * t.a * t.c;
 if( delta<0 )
   rozw.ilosc_rozwiazan=BRAK_ROZWIAZAN;
 else
   if( delta==0 )
     {
      rozw.ilosc_rozwiazan=JEDNO_ROZWIAZANIE;
      rozw.x1 = rozw.x2 = -t.b/(2*t.a);
     }
   else
     {
      rozw.ilosc_rozwiazan=DWA_ROZWIAZANIA;
      delta = sqrt(delta);
      rozw.x1 = (-t.b - delta)/(2*t.a);
      rozw.x2 = (-t.b + delta)/(2*t.a);
     }
 return rozw;
}


void main()
{
 T_ROZWIAZANIE_TROJMIANU t;
 t = Trojmian( Wczytaj3liczby("Podaj wspolczynniki trojmianu"));
 switch( t.ilosc_rozwiazan )
   {
    case TO_NIE_TROJMIAN:
         cout<<"To nie jest trojmian"; break;
    case BRAK_ROZWIAZAN:
         cout<<"Ten trojmian nie ma rozwiazan rzeczywistych";
         break;
    case JEDNO_ROZWIAZANIE:
    case DWA_ROZWIAZANIA:
         cout << "Ten trojmian ma rozwiazania rzeczywiste";
         break;
   };
}
```

```
/*   PRZYKŁAD 2.
     WYKORZYSTANIA TYPU STRUKTURALNEGO DO
     PRZECHOWYWANIA INFORMACJI O SPISIE / LIŚCIE ELEMENTÓW
     W POSTACI TABLICY STRUKTUR  */

#define DL_IMIENIA 15
typedef char T_IMIE[ DL_IMIENIA+1 ];


struct T_DANE_OSOBOWE
  {
   char   nazwisko[31];
   T_IMIE imie;     // == char imie[16];
   int    wiek;
  };


//Najprostsza definicja tablicy struktur
T_DANE_OSOBOWE  spis_osob_1[100];


//to samo z wykorzystaniem definicji nowego typu
const IL_OSOB=100;
typedef T_DANE_OSOBOWE  T_TABLICA_OSOB[ IL_OSOB ];
T_TABLICA_OSOB  spis_osob_2;


//przykład funkcji wyznaczającej adres danych najstarszej osoby
T_DANE_OSOBOWE* Najstarszy( T_DANE_OSOBOWE tab[] )
{
 //zmienne pomocnicze przechowująca wiek i pozycje najstarszej osoby
 int max_wiek=tab[0].wiek;
 int poz_max=0;
 for(int i=1; i<IL_OSOB; i++)
   if( tab[i].wiek > max_wiek )
     {
      max_wiek=tab[i].wiek;
      poz_max=i;
     }
 return &tab[poz_max];
}
```

```
/* "obiektowa" definicja tablicy (???)
    Połączenie w jedną całość tablicy i związanego z nią licznika
    ( ilość elementów tablicy wypełnionych danymi )
    jako pary pól o nazwach tab i ilosc
    oraz dodanie funkcji inicjującej wybrane pola podczas tworzenia struktury */

struct T_SPIS_OSOB
  {
   T_DANE_OSOBOWE  tab[ IL_OSOB ];
   unsigned        ilosc;

   //opcjonalna inicjacja zawartości (specjalna funkcja - konstruktor)
   T_SPIS_OSOB(){ilosc=0;}

      /*nazwa konstruktora jest taka sama jak nazwa struktury / klasy,
        może być zdefiniowanych kilka wersji konstruktora,
        ten powyżej to jest konstruktor bezparametrowy - domyślny */
  };

//inna wersja funkcji "Najstarszy"
T_DANE_OSOBOWE* Najstarszy_T( T_SPIS_OSOB sp )
{
 T_DANE_OSOBOWE* naj=sp.tab;   // = &sp.tab[0]
 for(int i=1; i<sp.ilosc; i++)
   if( sp.tab[i].wiek > (*naj).wiek )
     naj = &sp.tab[i];
 return naj;
}

void main()
{
 T_DANE_OSOBOWE* senior;

 //możliwe postacie definicji tablicy struktur
 T_DANE_OSOBOWE  grupa_1[100];   //definicja 1
 T_TABLICA_OSOB  grupa_2;        //definicja 2
 T_SPIS_OSOB     grupa_3;        //definicja 3

 //możliwe wywołania funkcji
 senior = Najstarszy( grupa_1 );
 senior = Najstarszy( grupa_2 );
 senior = Najstarszy( grupa_3.tab );
 senior = Najstarszy_T( grupa_3 );
}
```