TYP STRUKTURALNY

Struktury → najbardziej elastyczny sposób reprezentowania danych w języku C (odpowiednik rekordów w języku Pascal),

→ obiekt złożony z jednej lub kilku zmiennych, które mogą być różnego typu (w przeciwieństwie do tablic),

→ budowa - układ i typy pól składowych - typu strukturalnego są definiowane przez programistę.

Przykład:

nazwisko	imię	rok_urodz	płeć	wzrost	stypendium
char [30]	**char** [15]	**int**	**char**	**unsigned char**	**double**

- **Deklarowanie typu strukturalnego**

```
struct  nazwa_typu        ← nazwa tworzonego typu strukturalnego
   {
     typ_pola_1   nazwa_pola_1;
     typ_pola_2   nazwa_pola_2;  ← typy i nazwy pól składowych
       • • •
     typ_pola_n   nazwa_pola_n;
   } ;
```

Przykłady:

```
struct dane_osobowe
   {
     char  nazwisko[31];
     char  imie[16];
     int    rok_urodz;
     char  plec;
     unsigned char  wzrost;
     double  stypendium;
   } ;

struct punkt  {  double x, y;  } ;

struct okrag
   {
     struct punkt  srodek_okregu;          // struktura zagnieżdżona
     double  promien;
   } ;
```

- **Definiowanie (tworzenie) zmiennych strukturalnych**

struct nazwa_typu nazwa_tworzonej_zmiennej**;**

Przykład:

```
struct dane_osobowe   student;
struct okrag  figura_1;
```

- Można połączyć definicję zmiennej z inicjacją jej wartości. Np.

```
                                  // { Nazwisko, Imie, Rok_urodz, Plec,Wzrost, Stypendium }
struct dane_osobowe   student = { "Kowalski", "Jan", 1970,'M', 175, 320.0 };
                                        //  { { x ,  y } , promień }
struct okrag  figura_1 = {  { 10.0, 15.75 } , 50.5 };
```

- Można połączyć deklarację typu strukturalnego z definicją zmiennych

```
struct  nazwa_typu                    ← nazwę typu można pominąć
   {
     typ_pola_1   nazwa_pola_1;
     typ_pola_2   nazwa_pola_2;  ← typy i nazwy pól składowych
       • • •
     typ_pola_n   nazwa_pola_n;
   } nazwa_ zmiennej ;                ← nazwa definiowanej zmiennej
```

Np.

```
struct                                          // pominięto nazwę typu
  {
    char  nazwisko[31];
    char  imie[16];
    int    rok_urodz;
    char  plec;
    unsigned char  wzrost;
    double  stypendium;
  } student_1, student_2;          // definicja dwóch zmiennych strukturalnych

struct okrag                 // jednoczesna deklaracja typów i definicja zmiennych
  {
    struct punkt                 // bezpośrednia definicja struktury zagnieżdżonej
      {
        double x, y;
      }  srodek_okregu;
    double  promien;
  } figura_1;
```

- **<u>Odwoływanie się do elementów struktury</u>** $\rightarrow$ za pomocą kropki

Przykłady:

```
student . wzrost = 180 ;                          // przypisanie wartości
student_1 . wzrost = student_2 . wzrost;
figura_1 . promien = 50;
figura_1 . srodek_okregu . x = 15;
scanf( "%lf" , &student . stypendium );           // wczytanie z klawiatury
scanf( "%s" , student . nazwisko );               // wczytanie łańcucha znaków
strcpy( student . imie, "Tomasz" );
```

W pierwotnej wersji języka **C** (Kernigham, Ritchie) jedynymi dozwolonymi operacjami na strukturze były pobranie adresu (&) oraz działania na składowych.

W wersji ANSI-C (Turbo C++) możliwe jest bezpośrednie przypisanie struktur, struktura może być również argumentem i wynikiem zwracanym przez funkcję.

```
student_1 = student_2;                            // bezpośrednie przypisanie struktur
memcpy( &student_1, &student_2, sizeof( student_1 ) );
~~student_1 . nazwisko = student_2 . nazwisko~~
strcpy( student_1 . nazwisko, student_2 . nazwisko );

                                                  // funkcja zwracająca daną strukturalną
struct dane_osobowe   Wczytaj_Dane_Osobowe( void )
{
   struct dane_osobowe nowe_dane;
   printf( "Podaj nazwisko: " );
   scanf( "%s" , nowe_dane . nazwisko );
      • • •
   return( nowe_dane );
}

                                                  // funkcja której argumentem jest zmienna strukturalna
void Wyswietl_Dane_Osobowe( struct dane_osobowe  osoba )
{
   printf( "Nazwisko: %s\n" , osoba . nazwisko );
   printf( "       Imie: %s\n" , osoba . imie );
      • • •
}
```

- **Struktura jako element tablicy**

```
struct dane_osobowe  baza[ 100 ];                    // definicja tablicy struktur
struct dane_osobowe  mala_baza[ 2 ] =  {
                                  { ”Kowalski”, ”Jan”, 1970,‘M’, 175, 320.0 } ,
                                  { ”Nowak”, ”Tomasz”, 1965,‘M’, 180, 50.0 }
                                                  };
struct okrag  figury[ 3 ] = {  {{15, 10}, 50} ,  {{0, 0}, 10} ,   {{30, -70}, 8}  };

baza[ 2 ] . wzrost = 150;              // przykładowe operacje na tablicach struktur
figury[ 0 ] . srodek_okregu . y = 50;
for( int i = 0; i < 100; i++ )
   {
      printf( ”Podaj nazwisko: ” );
      scanf( ”%s” , baza[ i ] . nazwisko );
      • • •
      printf( ”Podaj stypendium: ” );
      scanf( ”%lf” , &(baza[ i ] . stypendium) );
   }
```

- **Wskaźniki do struktur** - dostęp do struktury za pośrednictwem adresu

```
struct dane_osobowe  student;
struct dane_osobowe  *wsk_os;
student . wzrost = 180;              // bezposrednie przypisanie do pola struktury
wsk_os = &student;
(*wsk_os) . wzrost = 180;            // pośrednie przypisanie poprzez wskaźnik
wsk_os –> wzrost = 180;                    // j.w. z nowym operatorem strzałki


                                   // funkcja porządkująca rosnąco zawartości dwu zmiennych
void Uporzadkuj(    struct dane_osobowe *wsk_os_1,
                    struct dane_osobowe *wsk_os_2   )
{  struct dane_osobowe bufor;
   if( wsk_os_1–>wiek > wsk_os_2–>wiek )
      {
         bufor = *wsk_os_1;
         *wsk_os_1 = *wsk_os_2;
         wsk_os_2 = bufor
      }
} //-------------------------------------------------------------------------Uporzadkuj
```