

TYP STRUKTURALNY

- Struktura → najbardziej elastyczny sposób reprezentowania danych w języku C (odpowiednik rekordu 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.

Definiowanie nowego 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;
};
```

Przykład:

Kowalski	Jan	1987	173 cm	'm'	437.20zł
nazwisko	imię	rok_urodz	wzrost	pleć	stypendium
char [30]	char [15]	short	short	char	double

```
struct T_dane_osobowe
{
    char    nazwisko[30];
    char    imie[15];
    short   rok_urodz, wzrost;
    char    plec;
    double  stypendium;
};

struct T_data_kalendarzowa { int dzien, miesiac, rok; };

struct T_pozyczka
{
    double    kwota;
    char      opis[50];
    T_data_kalendarzowa data_pozyczki;    // struktura zagnieżdżona
    T_data_kalendarzowa data_zwrotu;      // struktura zagnieżdżona
};
```

Definiowanie (tworzenie) zmiennych strukturalnych:

```
struct nazwa_typu_strukturalnego nazwa_tworzonej_zmiennej; // C
nazwa_typu_strukturalnego nazwa_tworzonej_zmiennej; // C++
```

Przykłady:

```
T_dane_osobowe student_1;
T_dane_osobowe student_2, student_3, nowy_student;
T_dane_osobowe grupa_studentow[ 20 ];

T_pozyczka pozyczka_dla_Tomka;
T_pozyczka moje_pozyczki[ 100 ];
```

Można połączyć definicję zmiennej z inicjalizacją jej wartości. np.

```
// { nazwisko, imie, rok_urodz, wzrost, plec, stypendium }
T_dane_osobowe student_x = {"Kowalski", "Jan", 1970, 175, 'M', 320.00 };

// { kwota, opis, { dzien, miesiac, rok }, { dzien, miesiac, rok } }
T_pozyczka ostatnia = { 100.00, "na obiady", {27,11,2006}, {3,12,2006}};
```

Można połączyć definicję 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[30];
    char imie[15];
    short rok_urodz, wzrost;
    char plec;
    double stypendium;
} student_1, student_2;                               // definicja dwóch zmiennych strukturalnych
```

Odwoływanie się do elementów struktury: za pomocą operatora kropki

Przykłady:

```
student_1.wzrost = 180 ;           // przypisanie wartości do pola struktury
student_2.wzrost = student_1.wzrost;

pożyczka_dla_Tomka.kwota = 50.00;
pożyczka_dla_Tomka.data_zwrotu.miesiac = 12;

scanf( "%lf" , &student.stypendium );           // wczytanie pola liczbowego
scanf( "%s" , student.nazwisko );             // wczytanie łańcucha znaków

cin >> student.stypendium;
cin >> student.nazwisko;

strcpy( student.imie, "Tomasz" );           // kopiowanie tekstu do pola struktury
```

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

W wersji **C++** możliwe jest bezpośrednie przypisanie struktur, struktura może być również argumentem i wynikiem zwracanym przez funkcję.

```
memcpy( &student_2, &student_1, sizeof(student_1) ); // w języku „C” !
student_2 = student_1; // bezpośrednio przypisanie struktur możliwe w „C++”

student_2.nazwisko = student_1.nazwisko;           // uwaga na teksty!
strcpy( student_2.nazwisko, student_1.nazwisko );
```

```
                                     // funkcja zwracająca daną strukturalną
T_dane_osobowe Wczytaj_Dane_Osobowe( void )
{
    T_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(const T_dane_osobowe &osoba )
{
    printf( "Nazwisko: %s\n" , osoba.nazwisko );
    printf( "    Imie: %s\n" , osoba.imie );
    • • •
}
```

Struktura jako element tablicy:

```
T_dane_osobowe pusta_baza[ 100 ];           // definicje tablic struktur

T_dane_osobowe baza[ 3 ] = {
    { "Kowalski", "Jan", 1970, 175, 'm', 95.00 } ,
    { "Nowak", "Tomasz", 1965, 180, 'm', 0.00 } ,
    { "Nowak", "Anna", 1983, 162, 'k', 250.0 } };

baza[0].wzrost = 175;           // przykładowe operacje na strukturze w tablicy
baza[1].nazwisko[0] = 'N';
strcpy( baza[2].imie, "Anna" );
```

	nazwisko	imie	rok_urodz	wzrost	plec	stypendium
0	Kowalski	Jan	1970	175	'm'	95.00
1	Nowak	Tomasz	1965	180	'm'	0.00
2	Nowak	Anna	1983	162	'k'	250.00

```
for( int i = 0; i < 3; i++ )           // wydrukowanie zawartości całej tablicy struktur
{
    printf( "Osoba numer [%d] \n" , i+1 );
    printf( "Nazwisko: %s \n" , baza[ i ].nazwisko );
    . . .
    printf( "Stypendium: %.2f \n" , baza[ i ].stypendium );
}
```

Wskaźniki do struktur: podczas dostępu do struktury za pośrednictwem adresu

```
T_dane_osobowe student;
student.wzrost = 180;           // bezpośrednio przypisanie do pola struktury

T_dane_osobowe *wsk_os;           // wskaźnik do struktury
wsk_os = &student;

(*wsk_os).wzrost = 180;           // pośrednie przypisanie poprzez wskaźnik
wsk_os->wzrost = 180;           // to samo z wykorzystaniem operatora strzałki

cin >> baza[ 2 ].stypendium;           // scanf("%lf",&(baza[2].stypendium));
cin >> (baza+2)->stypendium;           // scanf("%lf",&((baza+2)->stypendium));

for( wsk_os=baza; wsk_os<baza+3; wsk_os++)
    cin >> wsk_os->stypendium;
```

Przykład 1 : wczytywanie / wyświetlanie / filtrowanie listy pożyczek

```
#include <iostream>                                     // implementacja w języku C++

struct T_data_kalendarzowa { int dzien, miesiac, rok; };

struct T_pożyczka { double kwota; char opis[50];
                   T_data_kalendarzowa data_pożyczki, data_zwrotu;};

const N=3;
T_pożyczka spis[N];

int main()
{
    cout<<"Podaj dane do spisu pożyczek:";
    for(int i=0; i<N; i++) {
        cout<<"\n\nPożyczka nr "<< i <<endl;
        cout<<"Kwota = ";           cin>>spis[i].kwota;
        cout<<"Opis = ";           cin>>spis[i].opis;
        cout<<"Data pożyczki:\n";
        cout<<" dzien = ";         cin>>spis[i].data_pożyczki.dzien;
        cout<<" miesiac = ";       cin>>spis[i].data_pożyczki.miesiac;
        cout<<" rok = ";           cin>>spis[i].data_pożyczki.rok;
    }

    cout<<"\n\nWyświetlenie spisu pożyczek:";
    for(int i=0; i<N; i++) {
        cout<<"\n\nPożyczka nr "<< i <<endl;
        cout<<"Kwota = "<<spis[i].kwota<<" Opis = "<<spis[i].opis;
        cout<<"\nData pożyczki: "<<spis[i].data_pożyczki.dzien<<"/";
        cout<<spis[i].data_pożyczki.miesiac<<"/";
        cout<<spis[i].data_pożyczki.rok;
    }

    cout<<"\n\nLista dużych pożyczek:";
    for(int i=0; i<N; i++)
        if( spis[i].kwota>100 ) {
            cout<<"\n\nPożyczka nr "<< i <<endl;
            cout<<"Kwota = "<<spis[i].kwota<<" Opis = "<<spis[i].opis;
            cout<<"\nData pożyczki: "<<spis[i].data_pożyczki.dzien<<"/";
            cout<<spis[i].data_pożyczki.miesiac<<"/";
            cout<<spis[i].data_pożyczki.rok;
        }

    cout<<"\n\nKoniec programu. Nacisnij ENTER";
    cin.ignore(); cin.get();
    return 0;
}
```

Przykład 2 : inna wersja przykładu (1) → z wykorzystaniem funkcji

```
#include <iostream> // implementacja w języku C++

// --- definicja daty kalendarzowej oraz funkcji WCZYTAJ / WYSWIETL ---

struct T_data_kalendarzowa { int dzien, miesiac, rok; };

void WCZYTAJ_DATE(T_data_kalendarzowa& data)
{
    cout<<" dzien = ";    cin>>data.dzien;
    cout<<" miesiac = ";  cin>>data.miesiac;
    cout<<" rok = ";     cin>>data.rok;
}

void WYSWIETL_DATE(const T_data_kalendarzowa& data)
{
    cout<<data.dzien<<"/"<<data.miesiac<<"/"<<data.rok;
}

// ----- definicja pożyczki oraz funkcji WCZYTAJ / WYSWIETL -----

struct T_pożyczka { double kwota; char opis[50];
                  T_data_kalendarzowa data_pożyczki,data_zwrotu;};

void WCZYTAJ_POZYCZKE(T_pożyczka& pożyczka)
{
    cout<<"Kwota = ";    cin>>pożyczka.kwota;
    cout<<"Opis = ";    cin>>pożyczka.opis;
    cout<<"Data pożyczki:\n";
    WCZYTAJ_DATE( pożyczka.data_pożyczki );
    cout<<"Data zwrotu:\n";
    WCZYTAJ_DATE( pożyczka.data_zwrotu );
}

void WYSWIETL_POZYCZKE(const T_pożyczka& pożyczka)
{
    cout << "Kwota = " << pożyczka.kwota;
    cout << " Opis = " << pożyczka.opis;
    cout << "\nData pożyczki: ";
    WYSWIETL_DATE( pożyczka.data_pożyczki );
    cout << "\nData zwrotu: ";
    WYSWIETL_DATE( pożyczka.data_zwrotu );
}
```

c.d. przykładu (2)

```
// ----- Program główny -----  
// ----- Wykorzystujący zdefiniowane wcześniej struktury i funkcje -----  
  
// T_data_kalendarzowa / WCZYTAJ_DATE / WYSWIETL_DATE  
// T_pozyczka / WCZYTAJ_POZYCZKE / WYSWIETL_POZYCZKE  
  
const N=3;  
T_pozyczka spis[N];  
  
int main()  
{  
    cout<<"Podaj dane do spisu pozyczek:";  
    for(int i=0; i<N; i++)  
    {  
        cout<<"\n\nPozycja nr "<< i <<endl;  
        WCZYTAJ_POZYCZKE( spis[i] );  
    }  
  
    cout<<"\n\nWyswietlenie spisu pozyczek:";  
    for(int i=0; i<N; i++)  
    {  
        cout<<"\n\nPozycja nr "<<i<<endl; ~~~~~  
        WYSWIETL_POZYCZKE( spis[i] ); ~~~~~  
    }  
  
    cout<<"\n\nLista duzych pozyczek:";  
    for(int i=0; i<N; i++)  
        if( spis[i].kwota>100 ) ~~~~~  
        {  
            cout<<"\n\nPozycja nr "<< i <<endl; ~~~~~  
            WYSWIETL_POZYCZKE( spis[i] ); ~~~~~  
        }  
  
    cout<<"\n\nKoniec programu. Nacisnij ENTER";  
    cin.ignore(); cin.get();  
    return 0;  
}
```