7 Struktury

- Struktura to zbiór zmiennych tego samego lub różnego typu występujących pod wspólną nazwą. Można powiedzieć, że jest typem definiowanym przez użytkownika. Po zdefiniowaniu tego typu można tworzyć zmienne tego typu.
- Składnia deklaracji struktury:

```
struct etykieta_struktury
{
   typ_1 nazwa_składowej_1;
   typ_2 nazwa_składowej_2;
   ...
   typ_N nazwa_składowej_N;
};
```

gdzie:
  - *etykieta* identyfikuje strukturę.
  - dane struktury nazywane są *składowymi*, *polami* lub *elementami* (ang. *members*, *fields*, *elements*).

- Nazwy składowych danej struktury muszą być różne.
- Każda składowa struktury jest określonego typu (np. `int, float, char`),
- Deklaracja struktury kończy się średnikiem.

- Przykład 1a: struktura opisująca punkt; dwie zmienne typu strukturalnego, deklaracja struktury jest oddzielona od deklaracji zmiennych:

```
struct Punkt{        // etykieta struktury
 int x;              // składowe struktury
 int y;
 };

Punkt p1,p2;         // zmienne typu Punkt
```

Komentarz: w języku C++ do deklarowania zmiennych typu strukturalnego można używać jednej z dwóch składni:

```
Punkt p1,p2; // składnia C++, pominięte słowo struct
struct Punkt p1,p2; // w stylu języka C, użyte słowo struct
```

- Przykład 1b: zmienne zadeklarowane razem z deklaracją struktury:

```
struct Punkt{
   int x;
   int y;
} p1,p2;
```

- Przykład 2: struktura opisująca pracownika:

```
struct Osoba {
   int id;
   int wiek;
   float placa;
};

Osoba ksiegowy, sekretarka;
Osoba pracownicy[10]; /* deklaracja tablicy z elementami typu osoba */
```

7.1 Inicjowanie struktur

- Struktury inicjuje się podobnie do inicjacji tablic. Dopisuje się na końcu definicji listę wartości początkowych składowych struktury.

```
struct Punkt
{ int x;
  int y;
};
Punkt p1 = {1,1}; // Uwaga: nawiasy klamrowe!

struct Osoba {
   int id;
   int wiek;
   float placa;
};
Osoba pracownik = {1234, 25, 1200};

struct Ulamek {
   long Licznik,Mianownik;
} A, B, Zerowy={0,1}; // ułamki A i B nie mają jeszcze wartości
Ulamek C, D={1,4}, E={6,8}; // zainicjowanie tylko ułamków D i E
```

7.2 Dostęp do składowych struktury

- Aby odwołać się do składowej struktury, trzeba użyć *operatora wyboru składowej* oznaczanego *kropką*.
- Składnia:

nazwa_zmiennej_strukturowej.nazwa_składowej

- Przykład:

```
struct punkt
{ int x;
  int y;
};
punkt p1,p2;

// przypisywanie wartości
p1.x=1;
p1.y=1;

// wyświetlanie
cout << "Punkt p1(" << p1.x << ',' << p1.y << ')' << endl;

// wczytywanie
cin >> p2.x >> p2.y;

// użycie w wyrażeniu
odl=sqrt((double)p1.x*p1.x + (double)p2.y*p2.y);

struct Osoba{
   int id;
   int wiek;
   float placa;
};
Osoba pracownik;
cout << "Podaj identyfikator: ";
cin >> pracownik.id;
cout<< pracownik.id << endl;
```

7.3 Wzajemne przypisania struktur

- Dane zawarte w jednej strukturze można przypisać innej strukturze tego samego typu za pomocą operatora przypisania.
- Przykład (w4p1):

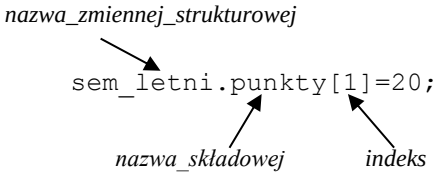
```
#include <iostream>
using namespace std;
struct punkt {
    int x;
    int y;
};
int main() {
    punkt p1, p2;
    p1.x=10;
    p1.y=20;
    cout <<"Wspolrzedne punktu p1: " << p1.x << ',' << p1.y << endl;
    p2=p1; // przypisanie jednej struktury innej
    cout <<"Wspolrzedne punktu p2: " << p2.x << ',' << p2.y << endl;
    return 0;
}
```

7.4 Struktury z tablicami

- Elementem struktury może być tablica.

```
struct oceny {
   long nr_indeksu;
   int ile_testow;
   int punkty[10];
   int ocena[10];
};
struct oceny sem_letni;
```

- Dostęp do elementu tablicy:



- Przykład (w4p2):

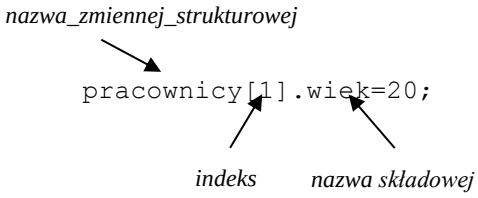
```
#include <iostream>
using namespace std;
struct Oceny {
   unsigned int indeks;
   int ile_testow;
   int punkty[10];
   int ocena[10];
};

int main(){
   Oceny sem_letni={1256,3,{28,15,10},{5,3,2}};
   cout << "Student nr indeksu: " << sem_letni.indeks
        << " Liczba testow:" << sem_letni.ile_testow << endl;
   for (int i=0;i<sem_letni.ile_testow;i++)
         cout << " Test nr " << (i+1) << " Punkty=" << sem_letni.punkty[i]
              << " Ocena=" << sem_letni.ocena[i] << endl;
   return 0;
}
```

7.5 Tablice struktur

- Zamiast wielu zmiennych można używać tablicy struktur.

```
struct Osoba {
   int id;
   float placa;
};
Osoba pracownicy[10];
```

- Dostęp do elementu tablicy:



- Przykład (w4p3):

```
#include <iostream>
using namespace std;
struct Osoba {
   int id;
   double placa;
};

int main()
{
   int i;
   int znak;
   const int ile=10;
   Osoba pracownicy[ile];

   for (i=0;i<ile;i++)
   {
      cout << "Wpisz identyfikator: ";
      cin >> pracownicy[i].id;
      cout << "Wpisz place: ";
      cin >> pracownicy[i].placa;
      while ( (znak=cin.get()) != '\n');
   }
   cout << "Identyfikator" <<'\t'<< "Placa" << endl;
   for (i=0;i<ile;i++)
      cout << pracownicy[i].id << '\t' <<
         pracownicy[i].placa << endl;
   return 0;
}
```

7.6 Struktury zawierające struktury

- Elementem składowym struktury może być inna struktura.

```
struct Data {
   int dzien; // data
   int miesiac;
   int rok;
};
struct Dzien {
   int dzien_tygodnia; // numer dnia w tygodniu 1 -poniedzialek
   Data data;
};

Dzien biezacy={1,{31,5,2001}};

cout << "Biezaca data: " << biezacy.data.dzien <<'.'
                         << biezacy.data.miesiac << '.'
                         << biezacy.data.rok
  << " Dzien tygodnia: " << biezacy.dzien_tygodnia << endl;

Dzien nastepny;

nastepny.dzien_tygodnia=2;
nastepny.data.dzien=1;
nastepny.data.miesiac=6;
nastepny.data.rok=2001;

cout << "Nastepna data: " << nastepny.data.dzien <<'.'
                          << nastepny.data.miesiac << '.'
                          << nastepny.data.rok
   << " Dzien tygodnia: " << nastepny.dzien_tygodnia << endl;
```

7.7 Przekazywanie struktur do funkcji przez wartość

Przekazywanie składowych struktury

```
#include <iostream>
using namespace std;

struct Ulamek {
   long licznik;
   long mianownik;
};

void DrukujUlamek(long x) {
   cout << x ;
}

int main() {
   struct Ulamek u;
   u.licznik=1; u.mianownik=2;
   cout << DrukujUlamek(u.licznik) << '/' << DrukujUlamek(u.mianownik);
   return 0;
}
```

Przekazywanie całych struktur

- Użycie struktury jako argumentu funkcji powoduje przekazanie **struktury** ***przez wartość*** **i** ***nie jest zalecane*** szczególnie w przypadku dużych struktur.

```
#include <iostream>
using namespace std;

struct Ulamek {
   long licznik;
   long mianownik;
};

void DrukujUlamek(char nazwa, Ulamek x) {
   cout << "Ulamek " << nazwa << ": "
        << x.licznik << '/' << x.mianownik << endl;
}

int main() {
   struct Ulamek u;
   u.licznik=1; u.mianownik=2;
   DrukujUlamek('u', u);
   cin.get();
   return 0;
}
```

7.8 Przekazywanie struktur do funkcji przez adres

- Do funkcji przekazujemy **adres struktury** a nie całą strukturę.

  Do pola adresu struktury odwołujemy się przez **->** zamiast kropki.

```
#include <iostream>
using namespace std;
struct Ulamek
{
    long licznik;
    long mianownik;
};

void DrukujUlamek(char nazwa, Ulamek *x) //adres x
{
    cout << "Ulamek " << nazwa << ": "
         << x->licznik << '/' << x->mianownik << endl;
    // x->licznik – pole licznik struktury o adresie x
}

bool CzytajUlamek(Ulamek *x)
{
    cout<<"Podaj ulamek ";
    cin>>x->licznik;
    cin>>x->mianownik;
    return (x->mianownik ==0 ? false : true);
}

Ulamek *innaCzytajU()  // wartością zwracaną z funkcji będzie adres x
{
    Ulamek x;
    cout<<"Podaj ulamek ";
    cin>>x.licznik;
    cin>>x.mianownik;

    if (x.mianownik==0)
    {
        cout<<"Mianownik nie może być 0"<<endl;
        return NULL;
    }
    return &x;  // zwracamy adres x
}

int main()
{
    Ulamek u, *u2;  // u2 bedzie przechowywalo adres struktury Ulamek
    u.licznik=1;
    u.mianownik=2;
    DrukujUlamek('u', &u); // do funkcji przekazujemy adres struktury u
   if(!CzytajUlamek(&u)){
     cout<<"Mianownik nie może być 0"<<endl;
     return -1;
        };
    DrukujUlamek('A',&u);

    u2 = innaCzytajU();
    if (u2!=NULL)
        DrukujUlamek('Z',u2);//nie potrzeba znaku & bo u2 to adres ulamka
    else
        return -1;
    return 0;

}
```

7.9 Zwracanie struktur z funkcji

- Można, ale niezalecane – lepiej zwracać adres struktury.
- Przykład

```
#include <iostream>
using namespace std;

struct Punkt {
int x;
int y;
};

Punkt wprowadz() {
    Punkt p;
    cout << "Podaj wspolrzedne punktu: ";
    cin >> p.x >> p.y;
    return p;
}
int main() {
    Punkt p1=wprowadz();
    cout << "Punkt p1: " << p1.x << ',' << p1.y << endl;

    Punkt p2=wprowadz();
    cout << "Punkt p2: " << p2.x << ',' << p2.y << endl;
    return 0;
}
```