Struktury

Jak wszyscy zapewne pamiętają, tablice przechowują wiele powiązanych ze sobą zmiennych jednego typu. Co jednak w sytuacji, w której z jakiegokolwiek powodu chcielibyśmy przechowywać kilka powiązanych ze sobą zmiennych różnych typów? Załóżmy, że chcielibyśmy przetwarzać w naszym programie informacje o samochodzie, jego markę, model, rocznik i pojemność silnika. Oczywiście możemy to zrobić w ten sposób:

```
int main(){
    string marka = "Citroen";
    string model = "C4 PIcasso";
    int rocznik = 2011;
    double silnik = 2.0;

    return 0;
}
```

Ale to niesie za sobą kilka konsekwencji – co musielibyśmy zrobić w sytuacji, w której mielibyśmy do opisania w programie więcej niż jeden samochód? Kolejne 4 zmienne, czyli z naszego kodu tworzymy śmietnisko. Dużo elegantszym rozwiązaniem będzie zastosowanie struktury. Struct pozwala w ramach jednej dużej zmiennej przechowywać więcej zmiennych, porządkując wszystko. Pozwoli też na deklarowanie wielu szerokich zmiennych o takiej samej strukturze:

```
struct{
        string marka;       //
        string model;       //
        int rocznik;        //
        double silnik;      // - deklaracja układu struktury
    } myVehicle;            // - deklaracja zmiennej

    myVehicle.marka="Citroen";
    myVehicle.model="C4 Picasso";
    myVehicle.rocznik=2011;
    myVehicle.silnik=2.0;
    cout<<myVehicle.marka<<" "<<myVehicle.model<<" "
<<myVehicle.rocznik<<" "<<myVehicle.silnik;
    return 0;
```

W powyższym przykładzie tworzymy schemat struktury, czyli określamy, jakie cechy będzie miała tworzona zmienna. W tym wypadku 2 stringi opisujące markę i model, integer określający rocznik i double przechowujący informację o pojemności silnika.

Deklarację struktury zakończyć należy średnikiem, ale przed nim umieściłem jeszcze nazwę zmiennej, do której się będę odnosił. To jest tzw. Unnamed struct. Stworzyliśmy teraz zmienną o nazwie myVehicle, która przechowuje 4 zmienne. Aby odwołać się konkretnie do tych pól musimy skorzystać z kropki – Jeśli chcemy odnieść się do pola „marka" struktury myVehicle napiszemy

```
myVehicle.marka="Citroen";
```

Możemy też zadeklarować wszystko naraz:

```
struct{
        string marka;       //
        string model;       //
        int rocznik;        //
        double silnik;      // - deklaracja układu struktury
    } myVehicle={"Citroen","C4 Picasso", 2011, 2.0};
```

Można tworzyć od razu kilka zmiennych, jeżeli jednej struktury potrzebować będziemy kilkukrotnie:

```
        double silnik;      // - deklaracja układu struktury
    } myVehicle, myVehicle2, myVehicle3;
```

Osobiście preferowaną przeze mnie opcją są struktury nazwane. W tym wypadku tworzymy de facto nowy typ danych, który składać się będzie z określonych w strukturze pól. W ten sposób, zamiast dla trzech pojazdów deklarować 12 zmiennych deklarujemy tylko 4:

```
struct car{
        string marka;       //
        string model;       //
        int rocznik;        //
        double silnik;      // - deklaracja układu struktury
    };

    car myVehicle={"Citroen","C4 Picasso", 2011, 2.0};
    car dadsVehicle={"Citroen", "C5 Aircross", 2019, 2.0};

    cout<<myVehicle.marka<<" "<<myVehicle.model<<"
"<<myVehicle.rocznik<<" "<<myVehicle.silnik<<endl;
    cout<<dadsVehicle.marka<<" "<<dadsVehicle.model<<"
"<<dadsVehicle.rocznik<<" "<<dadsVehicle.silnik<<endl;
```

Takie zastosowanie, czyli nowa zmienna pozwala nam pójść o kroczek dalej i utworzyć tablicę pojazdów

```
struct car{
        string owner;       //
        string make;       //
        string model;       //
        int year;        //
        double engine;      // – deklaracja układu struktury
    };

    car hylaVehicles[3] = {{"Michał", "Citroen","C4 Picasso",
2011, 2.0},
    {"Dad", "Citroen", "C5 Aircross", 2019, 2.0},
{"Mom","Toyota","Yaris", 2018, 1.5}};

    for (int i=0; i<3;i++){
        cout<<hylaVehicles[i].owner<<" "
<<hylaVehicles[i].make<<" "<<hylaVehicles[i].model<<" "
<<hylaVehicles[i].year<<" "<<hylaVehicles[i].engine<<" "
<<endl;
    }
    return 0;
```

Zadanie

Napisz program, który będzie przechowywał informacje o ocenach studentów z jednej grupy z przedmiotu PRG. Każdy student ma predefiniowane imię i nazwisko oraz swój numer indeksu. Program powinien zapytać użytkownika po kolei o ocenę każdego studenta, a następnie wydrukować zestawienie wszystkich ocen w formacie `nr_indeksu: ocena`

```
s14616: 2.0
s2137: 3.5
```

Oraz średnią wszystkich ocen za przedmiot. Studentów jest 10;