Co to jest klasa?

Najprościej mówiąc,
klasa jest to grupa danych i funkcji, które na tych danych operują.
Tworząc klasę tworzymy nowy typ danych, którego składnikami są inne typy danych.

Definicja klasy

Ogólny szablon definiowania klas wygląda tak:

```
class nazwa_klasy
   {
   ciało klasy (składowe,klasy);
   };
```

Oto definicja klasy przechowującej dane osobowe:

```
class osoba
   {
   public:
   string imie;
   string nazwisko;
   int wiek;
   };
```

Słowem kluczowym jest słówko **class**. Po nim następuje nazwa klasy. Dane składowe klasy zdefiniowane są między nawiasami klamrowymi { }. Definicja klasy kończy się średnikiem. Powyższa klasa ma tylko dane składowe (dwa łańcuchy znaków i zmienną typu int). Nie posiada ona funkcji składowych.

Definicja klasy składa się m.in. z deklaracji zmiennych różnego typu. Trzeba pamiętać, że są to tylko deklaracje, a nie definicje. Zmienne zadeklarowane w definicji klasy zostaną zdefiniowane (czyli zaistnieją w pamięci jako liczby) dopiero gdy stworzymy jakiś obiekt danej klasy. Oznacza to, że w definicji klasy nie można nadawać zmiennym wartości, ponieważ jeszcze nie istnieją.

Dostęp do składników klasy

Definiując klasę możemy ograniczyć dostęp do niektórych składników klasy.
Są trzy rodzaje dostępu do składników:

public
: oznacza, że składniki deklarowane po tej etykiecie są dostępne z każdego miejsca programu,

private
: oznacza, że składniki deklarowane po tej etykiecie dostępne są tylko dla funkcji składowych tej klasy; funkcje globalne (zwykłe) nie mają dostępu do tych składników, a więc z funkcji main również nie ma dostępu do tych danych,

protected

tak jak w przypadku private z tą tylko różnicą, że dostęp do takich składników mają jeszcze klasy wywodzące się z tej klasy (będzie to wytłumaczone przy dziedziczeniu)

W samym kodzie programu dostęp do zmiennych wewnątrz klasy, które są nazywane **polami** jest uzależniony od tego czy mamy do czynienia ze zmienną typu klasowego, czy wskaźnikiem na ten obiekt.

- W przypadku zmiennej o typie *klasa*, dostęp do pól uzyskuje się **operatorem wyłuskania**, którym jest kropka (**.**):

```
obiekt.pole = wartosc; //przypisanie wartości polu w obiekcie
```

- Jeżeli mamy do czynienia ze wskaźnikiem, **operatorem wyłuskania** jest strzałka **->** (myślnik i symbol większości):

```
obiekt->pole = wartosc;
```

Tworzenie obiektu klasy

Jak wspomniano wyżej sama definicja klasy nie tworzy jeszcze obiektu, który by przechowywał jakieś informacje. Aby tak było trzeba zadeklarować zmienną typu naszej klasy, lub utworzyć wskaźnik i przypisać mu taką zmienną (w dalszych przykładach będziemy się posługiwać tą drugą metodą).

Wskaźnik

Wskaźnik typu klasowego tworzy się używając operatora **new**:

```
osoba* czlowiek1 = new osoba;
```

taki wskaźnik trzeba przed zakończeniem programu zniszczyć operatorem **delete**:

```
delete czlowiek1;
```

W przeciwnym wypadku może wystąpić poważny błąd **wycieku zasobów**.

Zmienna

Zmienną typu klasowego tworzy się w ten sposób:

```
osoba czlowiek1;
```

Ta instrukcja tworzy nową **instancję** klasy *osoba* i przypisuje ją zmiennej *czlowiek1*.

Czym jest **instancja**?

Jest to kopia wszystkich pól klasy. Podczas tworzenia nowej instancji, tworzone są zmienne będące jej polami, oraz m.in. jest wywoływany jej konstruktor.
Jak to wygląda w praktyce, przedstawi poniższy przykład:

```
int main()
{
osoba czlowiek1;
osoba czlowiek2;//tworzone są dwa obiekty klasy "osoba"
czlowiek1.wiek = 20;//przypisanie pól
czlowiek2.wiek = 30;
cout << "Pierwszy człowiek ma " << czlowiek1.wiek << " lat." << endl
```

```
<< "Drugi człowiek ma " << czlowiek2.wiek << " lat." << endl;
return 0;
}
```

Co da w wyniku na ekranie:

```
Pierwszy człowiek ma 20 lat.
Drugi człowiek ma 30 lat.
```

Wynik może wydawać się oczywisty. Trzeba jednak nadmienić, że pole "wiek" w zmiennych *czlowiek1* i *czlowiek2* jest **zupełnie innym obiektem**. Dodatkowo można używać tylko pól nie zdefiniowanych w sekcjach **private** i **protected**

Funkcje w klasach

Oprócz pól klasa może posiadać także funkcje. Jest to główne kryterium odróżniające klasy od struktur.

Oto zmodyfikowana klasa *osoba*, z dodaną funkcją, oraz kwalifikatorem dostępu(**public**).

```
class osoba
  {
   public:
      string imie;
      string nazwisko;
      int wiek;
      void przedstawSie();//funkcja
  };
```

Funkcje z klas definiujemy później według poniższego wzoru (najlepiej po definicji klasy):

```
typ_zwracany nazwa_klasy::nazwa_funkcji(argumenty)
{...}
```

przykładowo funkcja *przedstawSie()* klasy *osoba* może brzmieć tak:

```
void osoba::przedstawSie()
{
 cout << "Nazywam się " << imie << " " << nazwisko << endl
 << "mam " << wiek << " lat." << endl;
}
```

Słówko **this** oznacza, że funkcja odwołuje się do pól klasy(a konkretniej jej przyszłej instancji), w której została zdefiniowana.

Teraz ponownie przykład z poprzedniego rozdziału tyle, że zmodyfikowany.

```
int main()
{
 osoba czlowiek1;//tworzenie obiektów
 osoba czlowiek2;
 czlowiek1.imie = "Jan";//wypełnianie pól pierwszego obiektu
 czlowiek1.nazwisko = "Kowalski";
 czlowiek1.wiek = 30;

 czlowiek2.imie = "Johan";//wypełnianie pól drugiego obiektu
 czlowiek2.nazwisko = "Smith";
 czlowiek2.wiek = 38;
 //a teraz wywołanie funkcji
```

```
  czlowiek1.przedstawSie();
  czlowiek2.przedstawSie();
  return 0;
 }
```

na ekranie powinniśmy ujrzeć:

```
Nazywam się Jan Kowalski
mam 30 lat.
Nazywam się Johan Smith
mam 38 lat.
```

W tym przykładzie widać również wyraźnie, że różne instancje mają swoje własne pola i funkcje działające na tych polach