Laboratorium nr 12

Temat: Struktury, klasy.

Zakres laboratorium:

- definiowanie struktur
- terminologia obiektowa
- definiowanie klas
- funkcje składowe klas
- programy złożone z wielu plików
- zadania laboratoryjne

Definiowanie struktur

Deklaracja struktury

```
struct Nasz_Typ;          //deklaracja struktury o nazwie Nasz_Typ
```

Definicja struktury

```
struct Nasz_Typ           //definicja struktury o nazwie Nasz_Typ
{
   //ciało struktury
   //…

};          //średnik!!!
```

Składniki struktury

```
struct Nasz_Typ
{
   //tutaj są składniki struktury, tj. zmienne różnych typów, także typów
   //zdefiniowanych przez użytkownika, np. obiektów innych struktur

   int skladnik1;
   float skladnik2;
   double skladnik3[20];
};
```

Definicja obiektów struktury

```
int zmienna;                    //definicja „zwykłej” zmiennej typu int
Nasz_Typ obiekt;                //definicja obiektu (zmiennej) typu Nasz_Typ
Nasz_Typ &referencja=obiekt;    //definicja referencji obiektu typu Nasz_Typ
Nasz_Typ *wskaznik=&obiekt;     //definicja wskaźnika do typu Nasz_Typ
```

Odnoszenie się do składników struktury

```
obiekt.skladnik
referencja.skladnik
wskaznik->skladnik
```

Przykład struktury – Czas

```
struct Czas                              //definicja struktury
{
   int godzina;
   int minuta;                           //dane składowe struktury
   int sekunda;
};
//-------------------------------------------------------------------------
void Wyswietl_Godzine(Czas cz)           //definicja funkcji globalnej
{
   cout<<cz.godzina<<":"<<cz.minuta<<":"<<cz.sekunda<<endl;
}
//-------------------------------------------------------------------------
main()
{
   Czas poczatek_zajec;                  //definicja obiektu struktury Czas

   poczatek_zajec.godzina=11;            //ustawianie składowych struktury
   poczatek_zajec.minuta=45;             //...
   poczatek_zajec.godzina=0;             //...

   Wyswietl_Godzine(poczatek_zajec);     //wywołanie funkcji globalnej

   Czas koniec_zajec;                    //definicja obiektu struktury Czas

   poczatek_zajec.godzina=13;
   poczatek_zajec.minuta=15;
   poczatek_zajec.godzina=0;

   Wyswietl_Godzine(koniec_zajec);
}
```

- Sama definicja struktury nie definiuje żadnych obiektów.
- Struktura to typ obiektu, a nie sam obiekt.
- W definicji struktury składniki **nie mogą być inicjalizowane**. Mogą im być nadawane wartości w ciele jakiejś funkcji, np. funkcji `main` lub funkcji „`ustaw`”.

```
struct Czas
{
   int godzina;
   int minuta=45;              //BŁĄD!!!
   int sekunda=0;              //BŁĄD!!!
};
```

- Dla każdego obiektu danej struktury istnieje w pamięci komputera osobny zestaw składników-danych tej struktury.

Terminologia obiektowa

Słownictwo obiektowe

Projektowanie zorientowane obiektowo (ang. object-oriented design, OOD) = **programowanie obiektowe** (ang. object-oriented programming, OOP) – metodyka tworzenia programów komputerowych, która definiuje programy za pomocą **obiektów**, czyli elementów łączących **stan** (**dane**) i **zachowanie** (**metody**, **funkcje**).

Obiektowy program – program komputerowy wyrażony jako **zbiór obiektów** komunikujących się pomiędzy sobą w celu wykonywania zadań.

Klasa – podstawowe pojęcie w programowaniu obiektowym. Klasy zawierają **dane** (**atrybuty**) oraz **funkcje** (**zachowania**), które są ze sobą ściśle związane. Na podstawie danej klasy mogą być wytwarzane konkretne **egzemplarze obiektów** (**obiekty**).
Klasa to **typ danych użytkownika (programisty)**. Dane klasy zwane są **danymi składowymi**, natomiast funkcje – **funkcjami składowymi** (lub **metodami**). Klasa to **typ obiektu**, a nie sam **obiekt**.

Obiekt – egzemplarz **typu zdefiniowanego przez użytkownika**, podobnie jak egzemplarz typu wbudowanego w język, np. `int` zwany jest **zmienną**.

Hermetyzacja (ukrywanie informacji) – inaczej niż obiekty, które doskonale wiedzą, w jaki sposób komunikować się między sobą za pomocą **interfejsu**, klienci klas nie powinni mieć informacji o tym, jak klasy zostały zaimplementowane – szczegóły implementacyjne ukryte są przed klientami korzystających z klas.

Definiowanie klas

Deklaracja klasy

```
class Nasz_Typ;      //deklaracja klasy o nazwie Nasz_Typ
```

Definicja klasy

```
class Nasz_Typ       //definicja klasy o nazwie Nasz_Typ
{
   //ciało klasy
   //……

};           //średnik!!!
```

Składniki klasy

```
class Nasz_Typ
{
   //tutaj są składniki klasy
   //tj. dane składowe klasy i funkcje składowe klasy
   //inne składniki klasy, np. obiekty innych klas, funkcje zaprzyjaźnione…

   int skladnik;
};
```

Definicja obiektów klasy

```
int zmienna;                    //definicja „zwykłej” zmiennej typu int
Nasz_Typ obiekt;                //definicja obiektu (zmiennej) typu Nasz_Typ
Nasz_Typ &referencja=obiekt;    //definicja referencji obiektu typu Nasz_Typ
Nasz_Typ *wskaznik=&obiekt;     //definicja wskaźnika do typu Nasz_Typ
```

Odnoszenie się do składników klasy

```
obiekt.skladnik
referencja.skladnik
wskaznik->skladnik
```

Zakres ważności składników klasy

Nazwy deklarowane w klasie mają zakres ważności równy **obszarowi całej klasy**.

Inaczej niż to było np. w zwykłych funkcjach, gdzie dana była znana od miejsca definicji aż do końca funkcji.

Enkapsulacja

Definicja klasy sprawia, że dane i funkcje składowe klasy nie są luźno rozrzucone w programie, ale są zamknięte jakby **w kapsule**. Do posługiwania się danymi składowymi klasy służą jej funkcje składowe (a także np. funkcje zaprzyjaźnione).

Etykiety dostępu do składników klasy

Istnieją etykiety, za pomocą których można określać **dostęp do składników klasy**.

- `private:`
- `protected:`
- `public:`

Składnik `private` – jest dostępny tylko dla funkcji składowych danej klasy (oraz dla funkcji zaprzyjaźnionych z tą klasą).

Składnik `protected` – jest dostępny tak, jak składnik `private`, ale dodatkowo jest jeszcze dostępny dla klas wywodzących się (**dziedziczących**) z danej klasy.

Składnik `public` – jest dostępny bez ograniczeń. Zwykle są to wybrane funkcje składowe, za pomocą których dokonuje się z zewnątrz operacji na danych prywatnych.

Przykład klasy – Pralka

```
class Pralka                                                    //definicja klasy
{
private:

   int nr_programu;
   int temperatura_prania;                                       //dane składowe
   int obroty_minute;                                            //klasy
   char nazwa_pralki[80];

   Zegar czas;                                                   //obiekty
   Silnik krokowy;                                               //innych klas

   void pranie_wstepne();                                        //funkcje narzędziowe
   void pranie_zasadnicze();                                     //klasy

   friend void serwis(Pralka &marka_pralki);                     //funkcje zaprzyjaźnione
   friend void dane_techniczne(Pralka *marka);                   //z daną klasą

public:

   void pranie(int program,int temperatura);
   void wirowanie(int minuty,int obroty);                        //funkcje składowe
   void plukanie();                                              //klasy
   bool start_stop();
};
//-------------------------------------------------------------------------

main()
{
   Pralka whirlpool,bosch;  //definicja dwóch obiektów klasy Pralka
}
```

- Sama definicja klasy nie definiuje żadnych obiektów.
- Klasa to typ obiektu, a nie sam obiekt.
- W definicji klasy składniki dane **nie mogą być inicjalizowane**. Mogą im być nadawane wartości za pomocą **konstruktora klasy** lub **funkcji składowych klasy**, np. funkcji „ustaw”.

```
class Pralka
{
private:

   int nr_programu;
   int temperatura_prania=60;          //BŁĄD!!!
   int obroty_minute=1000;             //BŁĄD!!!
};
```

- Dla każdego obiektu danej klasy istnieje w pamięci komputera osobny zestaw składników danych tej klasy.
- Funkcje składowe są w pamięci tylko jednokrotnie.
- Funkcje składowe **mają pełny dostęp** do wszystkich składników swojej klasy, tj. i do **danych** (mogą z nich korzystać i je modyfikować), i do innych **funkcji składowych** (mogą je wywoływać).

Funkcje składowe klas

- Funkcja składowa jest narzędziem, za pomocą którego dokonujemy operacji na danych składowych klasy, zwłaszcza na składnikach **private**, które są niedostępne spoza klasy.

- Funkcje publiczne (**public**) implementują zachowania i usługi, które klasa oferuje swoim klientom. Funkcje publiczne są zwykle zwane **interfejsem klasy** lub **interfejsem publicznym**.

- Funkcję składową wywołuje się **na rzecz konkretnego obiektu** klasy, tj.:

```
obiekt.funkcja(argumenty);
```

Przykład:

```
class Pralka
{
   int nr_programu;
   int temperatura_prania;
public:
   void pranie(int program,int temperatura);
   void plukanie();
};

main()
{
   Pralka whirlpool,bosch;                    //definicja dwóch obiektów
   whirlpool.pranie(nr_progr,temp_prania); //wywołanie funkcji na rzecz obiektu
   bosch.plukanie();                          //wywołanie funkcji na rzecz obiektu
}
```

Funkcja składowa może być zdefiniowana:

- wewnątrz samej definicji klasy (jest wtedy funkcją tzw. **inline**),
- poza ciałem klasy (wewnątrz klasy jest tylko deklaracja).

Przykład:

```
class Pralka
{
   int nr_programu;
   int temperatura_prania;
public:
   void pranie(int program,int temperatura);
   void plukanie()
   {
      //ta funkcja jest funkcją inline
      //w ten sposób należy definiować tylko bardzo krótkie funkcje
   }
};
//-------------------------------------------------------
void Pralka::pranie(int program,int temperatura)
{
   //funkcja zdefiniowana poza ciałem klasy
   //...
}
```

Programy złożone z wielu plików

- Każda definicja klasy jest umieszczana w osobnym **pliku nagłówkowym** (ang. *header file*, z rozszerzeniem **`.h`**).

- Definicje funkcji składowych poszczególnych klas umieszczane są w osobnych **plikach kodu źródłowego** (ang. *source-code file*, z rozszerzeniem **`.cpp`**) o tej samej podstawowej części nazwy, co plik nagłówkowy.

- Funkcja główna programu (`main()`) umieszczana jest we właściwym **pliku programowym** (z rozszerzeniem **`.cpp`**).

- Pliki nagłówkowe są włączane (za pomocą dyrektywy **`#include "nazwa.h"`**) do każdego pliku kodu źródłowego, w którym wykorzystywana jest dana klasa.

- Pliki z kodami źródłowymi są kompilowane i łączone z głównym programem.

```
//////////////////////////// Pralka.h – plik nagłówkowy ////////////////////////////////////
class Pralka
{
   int nr_programu;
   int temperatura_prania;
public:
   void pranie(int program,int temperatura);
   void plukanie();
};

//////////////////////////// Pralka.cpp – plik źródłowy ////////////////////////////////////
#include ”Pralka.h”
void Pralka::pranie(int program,int temperatura)
{
   //funkcja prania
   //...
}
//-------------------------------------------------------------------------------
void Pralka::plukanie()
{
   //funkcja płukania
   //...
}

//////////////////////////// Main.cpp – plik programowy ////////////////////////////////////
#include ”Pralka.h”
main()
{
   Pralka whirlpool,bosch;                  //definicja dwóch obiektów
   whirlpool.pranie(nr_progr,temp_prania);
   bosch.plukanie();
}
```

Zadania laboratoryjne