Laboratorium nr 6

Temat: Reguły zasięgu, zasłanianie nazw, obszary nazw.

Zakres laboratorium:

- zakres ważności nazwy obiektu, a czas życia obiektu
- reguły zasięgu
- zasłanianie nazw, operator rozróżniania zasięgu ::
- obszary nazw
- zadania laboratoryjne

Zakres ważności nazwy obiektu, a czas życia obiektu

Zakres ważności nazwy obiektu – to ta część programu, w której nazwa jest znana kompilatorowi.

Czas życia obiektu – to okres od momentu, gdy zostaje on zdefiniowany (definicja przydziela mu miejsce w pamięci) – do momentu, gdy przestaje on istnieć (jego miejsce w pamięci zostaje zwolnione).

Jaka jest różnica?

Różnica między powyższymi pojęciami jest taka, że w jakimś momencie obiekt może istnieć, ale nie być dostępny. To dlatego, że np. znajdujemy się chwilowo poza zakresem ważności jego nazwy.

Reguły zasięgu

Reguły zasięgu

Zasięg pliku – identyfikator jest dostępny we wszystkich funkcjach od miejsca, w którym został zadeklarowany, aż do końca pliku.
Przykład: wszystkie zmienne globalne, definicje funkcji i jej prototypy umieszczone poza funkcją.

Zasięg funkcji – jedynymi identyfikatorami mającymi zasięg funkcji są etykiety. Etykiety mogą być używane gdziekolwiek w funkcji, w której się pojawiają, ale nie są dostępne spoza ciała tej funkcji
Przykład: etykiety w strukturach `switch` (jako etykiety `case`) i etykiety w wyrażeniach `goto`.

Zasięg prototypu funkcji – jedynymi identyfikatorami mającymi zasięg prototypu funkcji są identyfikatory użyte na jej liście parametrów. Prototypy funkcji nie wymagają nazw na liście parametrów – wymagane są tylko typy.

Zasięg bloku – identyfikatory zadeklarowane wewnątrz bloku. Zasięg bloku rozpoczyna się w miejscu deklaracji identyfikatora i kończy w momencie napotkania prawego nawiasu klamrowego }.
Przykład: zmienne lokalne w funkcji, parametry funkcji, dowolne zmienne w bloku {}, itp.

Zasięg klasy – omówiony zostanie podczas omawiania klas.

ZALECENIE: Staraj się unikać korzystania z takich samych nazw identyfikatorów (zmiennych) w programie!!! (problemy z „zasłanianiem” nazw)

Zasłanianie nazw, operator rozróżniania zasięgu ::

Przykład:

```
#include <iostream>

using namespace std;

int zmienna=1;                    //definicja zmiennej globalnej

main()
{
   cout<<zmienna<<endl;
   {                              //otwarcie bloku lokalnego
      int zmienna=2;              //definicja zmiennej lokalnej

      cout<<zmienna<<endl;        //zmienna lokalna zasłania globalną
      cout<<::zmienna<<endl;      //tylko dla zasłoniętego obiektu globalnego
   }                              //zamknięcie bloku
   cout<<"Poza blokiem, "<<endl;
}
```

Widok ekranu:

```
1
2
1
Poza blokiem, 1
```

Obszary nazw

Obszary nazw (relatywnie nowa cecha C++) są przeznaczone dla programistów do pomocy w rozwijaniu nowych elementów programu bez wywoływania konfliktów nazw z istniejącymi elementami oprogramowania.

Każdy **plik nagłówkowy** w projekcie standardu C++ używa **obszaru nazw** nazywanego `std`. Projektanci nie powinni używać obszaru `std` do definiowania nowych bibliotek klas.

Instrukcja **`using namespace`** `std` mówi nam, że używamy elementów oprogramowania z biblioteki standardowej C++.

Instrukcja **`using`** umożliwia nam używanie krótkich wersji każdej nazwy w bibliotece standardowej C++ lub jakimkolwiek, określonym przez programistę **obszarze nazw `namespace`**.

Jeżeli używamy dwóch lub więcej bibliotek klas, które mają opisy z identycznymi nazwami, może wystąpić konflikt nazw. Należy wtedy w pełni określić nazwę, jakiej chcemy użyć z jej obszarem nazw za pomocą **operatora rozróżniania zasięgu ::**.

Przykład:

```
#include <iostream>

namespace student1
{
   int zmienna=1;
   int zmienna1=1;
}
////////////////////////////////////////////////////////////////////////////////
////////
namespace student2
{
   int zmienna=2;
   int zmienna2=2;
}

using namespace std;
using namespace student1;
using namespace student2;

main()
{

   cout<<student1::zmienna<<endl;
   cout<<student2::zmienna<<endl;

   cout<<zmienna1<<endl;
   cout<<zmienna2<<endl;
}
```

Zadania laboratoryjne