1. Czym jest kompilator i linker?

Kompilator to program, który czyta kod napisany w jednym języku (języku źródłowym) i tłumaczy go na równoważny kod w innym języku (języku wynikowym) z jednoczesnym wykrywaniem ewentualnych błędów popełnionych w trakcie programowania. Linker (konsolidator) to program łączący w trakcie procesu konsolidacji pliki obiektów i biblioteki kodów programu tworząc aplikacje wykonywalną.

2. Co to jest "include guard" lub inaczej "header guard"?

Include guard to technika pozwalająca na upewnienie się, że każdy plik zawierający definicję jakiegokolwiek elementu będzie przetwarzany tylko raz, tak, aby nie powodować błedu powtórzenia tego samego kodu.

3. Do czego służy #pragma once?

Dyrektywa #pragma once zapobiega ponownemu załączeniu treści całego pliku, w którym została użyta. Jednak jej ewentualna obsługa jest rozszerzeniem wprowadzonym przez dany kompilator i nie jest przenośna pomiędzy różnymi narzędziami.

4. Do czego służy słowo kluczowe "auto"?

W standardzie C++11 słowo kluczowe "auto" oznacza zastępczy typ zmiennej, który zostanie wydedukowany na podstawie wartości za pomocą której zmienna zostanie zainicjalizowana. Zmienna, której nadano zastępczy typ "auto" musi zostać zainicjalizowana w chwilii jej tworzenia. W przeciwnym wypadku zostanie zwrócony błąd kompilacji.

5. Omów na czym polega programowanie obiektowe?

W programowaniu obiektowym grupujemy w pewnym sensie dane w jednym module oraz dodajemy do nich metody postępowania z nimi - funkcje składowe. Powstaje w ten sposób moduł nie tylko kryjący dane, ale tównież nowy typ danej. Są to ukryte w danym module dane, oraz możliwe działanie na nich. Dane zgrupowane w poszczególny moduł stają się obiektem, możemy działać na nich, ale również nadawać im w pewien sposób uprawnienia (np. private, public), by uniemożlić osobom z zewnątrz dostania się do nich.

6. Omów wpływ modyfikatorów dostępu użytych podczas dziedziczenia na widoczność pól i metod.

Programując obiektowo programista podaje modyfikator dostępu dla poszczególnych pól i metod, by w ten sposób nie tylko wpłynąć na ich widoczność na zewnatrz klasy, ale również podczas dziedziczenia. W standardzie C++ znajdują się trzy modyfikatory dostępu, tj. public, private lub protected. Jeśli w klasie bazowej nasze pola i metody będą określone poprzez modyfikator "public", wtedy na poziomie klasy pochodnej programista będzie miał do nich swobodny dostęp, w przypadku dziedziczenia "public". Natomiast jeśli podczas dziedziczenia, na liście pochodzenia poprzedzimy klasę bazową modyfikatorem "private", wtedy wszystkie jej składniki dla klasy pochodnej również będą "private", to samo obowiązuje w przypadku dziedziczenia "protected". Używając słowa "private" dla wszystkich składników klasy bazowej (tj. pól i metod), dla każdej klasy pochodnej, bez względu na słowo poprzedzające listę pochodzenia, nigdy nie będzie dostępu do tych metod i pól, ale są one dziedziczone mimo wszystko. Jeśli natomiast wszystkie składniki klasy bazowej są określone mianem "protected", wtedy dla klasy pochodnej w przypadku dziedziczenia "public" są one publiczne z poziomu tej klasy. Natomiast w przypadku dziedziczenia "protected" są one również "protected" oraz w przypadku "private" stają się prywatne na poziomie klasy pochodnej.

7. Czym różni się metoda "virtual" od "pure virtual"?

Metodę "virtual" deklarujemy w klasach bazowych, by w przypadku dziedziczenia i nienadpisania danej metody w klasie pochodnej program mógł odnieść się do metody wirtualnej z klasy bazowej. Natomiast metoda "pure virtual" jest czysto wurtualna i pozwala określić klase bazową jako klasę abstrakcyjną, w przypadku której to wszystkie metody z niej muszą zostać nadpisane w klasach bazowych. Metoda ta również, w przeciwieństwie do metody "virtual" zostaje przyrównana przez programistę do zera, tj. =0.

8. Co to jest "vtable"?

Vtable to tablica funkcji wirtualnych. Występuje ona tylko w jednej kopii na całą klasę i zawiera tyle elementów, ile funkcji wirtualnych klasa ta posiada. Jej elementami są adresy w pamięci tych właśnie funkcji: pierwsza funkcja wirtualna ma więc adres zapisany w elemencie o indeksie 0 itd.

9. Opisz różnicę pomiędzy destruktorem a wirtualnym destruktorem. Jaki należy stosować częściej i dlaczego?

Destruktor jest metodą klasy wywoływaną podczas niszczenia obiektu danej klasy. Służy ona przede wszystkim do zwolnienia pamięci zarezerwowanej dla pól dynamicznych tej klasy. Destruktor wirtualny różni się od zwykłego destruktora tym, iż jest od wykorzystywany w klasie bazowej w przypadku dziedziczenia, by zaznaczyć zwolnienie pamięci i zniszczenie wszystkich obiektów pochodzących od klas pochodnych. W przypadku prostych programów, z niewielką ilością klas, bez dziedziczenia, dużo lepiej sprawdza się zwykły destruktor. Natomiast w przyadpku klas abstrakcyjnych, po których inne klasy będą dziedziczyć wskazane jest zastosowanie wirtualnego destruktora, by zniszczyć wszystkie obiekty z klas pochodnych.

Eliza Zych