* **Tip tanımlamalar bellekten yer kaplamaz, Nesne tanımlamaları kaplar.**

***Tip Tanimlamasi****:*

*class sinif*

*{*

*public:*

*void sinif::altsinif();*

*};*

***Nesne Tanimlamasi****:*

*Sinif snf;*

* **Bir sınıfa, constructor aracılığıyla argüman göndermek mümkündür.**

*myclass(int x); // constructor*

*void show();*

*};*

*myclass::myclass(int x)*

*{}*

*…*

*int main()*

*{*

*myclass ob(4);*

*…*

* Enum tanımlaması

*enum bisey {var, yok};*

*enum bisey Ahmet;*

*Ahmet=var;*

* **Class & Class’ın fonksiyonlarının bellek kullanımı hakkında:**

*Bir class’ın altında çalışırken o sırada yapılan işlemler hangi nesne tarafından gönderilmişse ona ayrılan bellekte iş görür. Yani her nesnenin kendine has fonksiyonları ve değişkenleri kendine has belleklerde tutulur.*

*Kopyalanan classlar ( class1 =class2) bellekte farklı yerlerde saklanır. Ama bit bit aynıdır. Böyle durumlarda sınıfın içerisinde dinamik ayrılmış bellek varsa, içlerinden birinin bu belleği free etmesi durumunda, diğerinin de kullanamayacağı göz önünde bulundurulmalıdır.*

* **Free() : Fonksiyon return ‘ünde (sonunda) mevcut automatic allocationlar (localler vs), free() ile dynamic allocationlar serbest bırakılır. Static allocationlar (static, global variable) serbest bırakılmaz. Serbest bırakılan bir nesnenin deconstructor’ı çalıştırılır.**
* **Friend fonksiyonlar ile bir veya birden fazla sınıfın private ‘ına ulaşılabilir.(Facebook gibi)**
* **Forward declaration; henüz sınıf tanımı yapılmamış olan bir sınıfı (compiler ‘da) kullanacaksak bu işlemden önce class sinif; şeklinde tanımlamamız gerekir.**
* **A reference (&) as another name for the same variable.**
  + *Referans iki kere assign edilemez, pointer edilebilir*
  + *Referans ‘ın bellekteki yerini öğrenemeyiz ama pointerin’ki bilinebilir.*
  + *Pointer can be assigned NULL directly, whereas reference cannot*
* **Fonksiyonlarda mandatory ve optional parametre verilebilir, overload fonksiyon yerine bu kullanım tercih edilebilir.**
  + *Void func( int x, int y=0) burada x: mandatory, y optional.*
* **Operator overloading :** 
  + *type sinif:operator# (type var) şeklinde bildirilir. Operatörün solunda kalan çağırıcıdır, yani this içerisinde gönderilir. Sağ tarafta kalan ise fonksiyon parametresi olarak gönderilir.*
  + *Tekli operatörlerde (++ob ya da ob++) overload fonksiyonu varsa operatörün sağda ya da solda olması önem kazanır. Bu overload fonksiyon parametre ister. Böyle bir overload fonksiyonu kullanabilmek için gönderilecek parametrenin sağda olmasına dikkat edilir.*

*coord coord::operator++()*

*{*

*x++;*

*y++;*

*return \*this;*

*}*

*//aşağıdaki olduğundan operatör sağdaysa artık notused //variable’ı solda gönderilmiş gibi olacak.*

*coord coord::operator++(int notused)*

*{*

*x=x+notused;*

*y=y+notused;*

*return \*this;*

*}*

* + *Friend functionlara this operatörü gönderilmediği için, gelen değerler tanımlanarak operatörün sağda veya solda olmasının önemi kaldırılır.*
  + *Fonksiyona & ile adres gönderilerek iş sağlamlaştırılır.*
* **Inheritance:**
  + *Tanım -> class sinifadi : erisimtürü baseclassadi() {}*
  + *Erişim türü: public ise base class’ın public’i türetilenin de publici olur. Protected’ı ise türetilenin de protected’i olur*
  + *Erişim türü: private ise baseclass’ın public’i ve Protected’i türetilenin private’ı olur.*
  + *Erişim türü:protected ise baseclass’ın public’i ve Protected’i türetilenin protected’i olur.*
* **Virtual Functions:** 
  + *Bir base sınıfta tanımlanır ve virtual konur basına.Türetilen sınıflarda aynı isim ile kullanılır(başına virtual konmadan). İçeriği her sınıfa özgüdür. Virtual func() {}=0 seklinde tanımlanırsa türetilen sınıflarda bu fonksiyonun tanımlama zorunluluğu olur.*