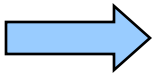
OBJEK LANJUT

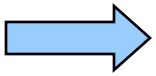
Oleh :
Rasim
Ilkom UPI

Objek Lokal & Global

```
Class ABC{
   char nama[20];
   public:
      ABC(char *nama);
      ~ABC();
};

ABC x("x");
Void main(){
   cout << "Awal fungsi main " << endl;
   ABC y("y");
   ABC z("z");
   cout << "Akhir fungsi main" << endl;
}
```



Objek Global



Objek Lokal

```
ABC ::ABC(char *nama){

   strcpy(ABC::nama,nama);
   cout <<"konstruktor " cout
   << ABC::nama
   << "dijalankan…" << endl;
}

ABC::~ABC(){
   cout<< "Destruktor " <<
nama
   << "dijalankan " <<endl;
}
```

Hasilnya adalah??

Objek Lokal & Global..2

```
Class ABC{
   char nama[20];
   public:
      ABC(char *nama);
      ~ABC();
};

ABC x(‘x”);

Void main(){
   cout << “Awal fungsi main “ << endl;
   ABC y(“y”);
   ABC z(“z”);
   exit(1);
}
```

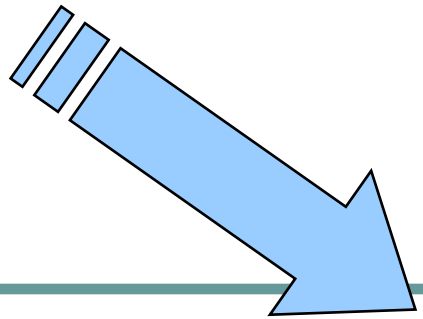
```
ABC ::ABC(char *nama){

   strcpy(ABC::nama,nama);
   cout <<“konstruktor “ cout
   << ABC::nama
   << “dijalankan…” << endl;
}

ABC::ABC(){
   cout<< “Destruktor “ <<
nama
   << “dijalankan “ <<endl;
}
```

Hasilnya adalah??

Menyebabkan main tidak selesai

OOP dengan ADT

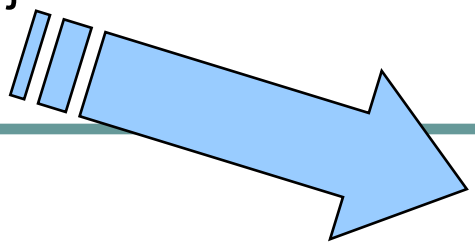
```
/* buku.h */
#ifndef __buku
#define __buku

Class Buku{
   char judul[35];
   char pengarang[25];
   int jumlah;

   public:
      Buku(char *judul="judul", char *pengarang="Pengarang",
      int jumlah=0);
      void info();
};
#endif
```



Body…….

OOP dengan ADT..2

```
/* buku.cpp */
#include <iostream.h>
#include <string.h>
#include “buku.h”

Buku(char *judul=“judul”, char *pengarang=“Pengarang”, int jumlah=0){
   strcpy(Buku::judul,judul);
   strcpy(Buku::pengarang,pengarang);
   Buku::jumlah = 0;
}
void info(){
    cout << “Judul          : “ << judul << endl;
    cout << “Pengarang  : “ << pengarang << endl;
    cout << “Jumlah       : “ << jumlah << ‘\n’ << endl;
}
```



Driver…….

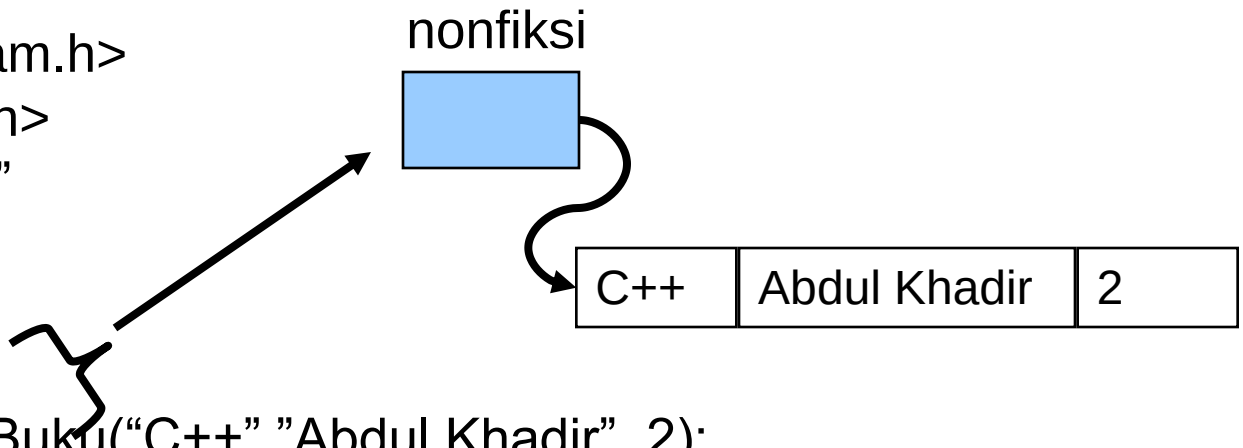
OOP dengan ADT..3

```
/* testbuku.cpp */
#include <iostream.h>
#include <conio.h>
#include “buku.h”

void main(){
    Buku literatur;
    Buku novel(“The Eagle Has Flown”, “Jack Hinggis”, 1);
    literatur.info();
    novel.info();
}
```

Hasilnya adalah??

Pointer ke Objek & Objek Dinamis

```
/* ptrobjek */
#include <iostream.h>
#include <conio.h>
#include “buku.h”

void main(){
   Buku *nonfiksi;
   nonfiksi =new Buku(“C++”,”Abdul Khadir”, 2);
   nonfiksi→info();
}
```



Nonfiksi→info() sama dengan (*nonfiksi).info();

Objek Sebagai Parameter

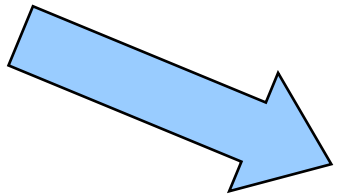
Melewatkan objek sebagai parameter dapat dilakukan dengan:

- Melewatkan objek berdasar nilai
- Melewatkan objek sebagai pointer
- Melewatkan objek sebagai referensi

Melewatkan objek berdasar nilai

Definisi prosedur:

```
Void info_buku(Buku b){
   cout << “informasi buku “ << ‘\n’ << endl;
   b.info();
}
```



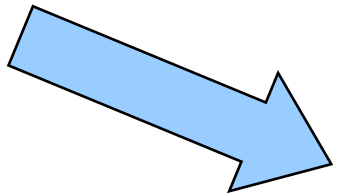
Pemanggilan:

```
Buku novel;
Info_buku(novel);
```

Melewatkan objek berdasar Pointer

Definisi prosedur:

```
Void info_buku(Buku *b){
    cout << “informasi buku “ << ‘\n’ <<
endl;
    (*b).info();     /* atau b→info();   */
}
```



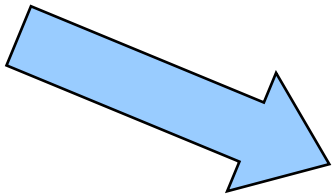
Pemanggilan:

```
Buku novel;
Info_buku(&novel);
```

Melewatkan objek berdasar Referensi

Definisi prosedur:

```
Void info_buku(Buku &b){
    cout << “informasi buku “ << ‘\n’ << endl;
     b.info();
}
```



Pemanggilan:

```
Buku novel;
Info_buku(novel);
```

NILAI BALIK BERUPA OBJECT

```
Class Buah {
        int apel;
        int jeruk;
    public:
      Buah(int jum_apel=0, int jml_jeruk=0);
      void info_buah();
      Buah tambah(buah b2);
};
```

NILAI BALIK BERUPA OBJECT..2

```
void main(){
        Buah buah1(20, 5);
        Buah buah2(12, 4);
        Buah buah3;

        buah3 = buah1.tambah(buah2);

       cout<<“objek buah 1” << endl;
       buah1.info_buah();

      cout<<“objek buah 2” << endl;
       buah2.info_buah();

      cout<<“objek buah 3” << endl;
       buah3.info_buah();
}
```

```
Buah::Buah(int jum_apel, int jum_jeruk){
     apel =jum_apel;
     jeruk=jum_jeruk;
}

void Buah::info(){
     cout<<“Jumlah apel = “ << apel
     << “ Jeruk = “ << jeruk << ‘\n’
     << endl;
}

Buah Buah::tambah(Buah b2){
   Buah tmp;
   tmp.apel = apel+b2.apel;
   tmp.jeruk = jeruk + b2.jeruk;
  return(tmp);
}
```

Data Anggota Bersifat Statis

```
class  klasstatis{
    public:
      statis int x;

      kelasstatis() {
           x++;
     }
    void info(){
       cout << “x = “ << x << endl;
   }
};
```

Data Anggota Bersifat Statis..2

```
Klasstatis::x=0; /* inisialisasi */
Void main(){
    klasststis x,y,z;
    x.info();
    y.info();
    z.info();
    kelasstatis w;
    w.info();
    cout << “Jumlah Class berklas statis\n”
    << yang telah diciptakan = “
   << klasstatis::x << “ buah” << endl;
   }
```

Fungsi Anggota Bersifat Statis

```
class  klasstatis{
    public:
      statis int x;

      kelasstatis() {
            x++;
       }
    void info(){
       cout << “x = “ << x << endl;
    }
    statis int nilai_x(){ return (x); }
};
```

Fungsi Anggota Bersifat Statis..2

```
Klasstatis::x=0; /* inisialisasi */
Void main(){
    klasststis x,y,z;
    x.info();
    y.info();
    z.info();
    kelasstatis w;
    w.info();
    cout << “Jumlah Class berklas statis\n”
    << yang telah diciptakan = “
   << klasstatis::nilai_x << “ buah” << endl;
   }
```

Pointer ke Fungsi Anggota

Sintak :

```
tipe(nama_class::*namapointer)(parameter,…);
```

Jika nama_class mempunyai nilai balik void, maka

```
void(nama_class::*namapointer)(void);
```

Pernyataan untuk menunjuk ke fungsi anggota:

```
Namapointer=&nama_class::fungsi_anggota
```

Contoh

```
class Klassx{
        public:
          void fung_1(){
              cout<<“Fung 1 dijalankan “ << endl;
          }

          void fung_2(){
            cout<<“Fung 2 dijalankan “ << endl;
          }

         void fung_3( int a, int b){
            return(a+b);
          }
};
```

Contoh..2

```
void main(){
      void (klassx::*ptr_fungsi) (void);
      klassx x;
      cout<< “Via fungsi anggota “ << endl;
      x.fung_1();
      x.fung_2();
      cout<<“5+8”<<x.fung_3(5,8) << endl;

      cout<<“\n via pointer ke fungsi anggota “ << endl;
      ptr_fungsi=&klassx::fung_1;
      (x.*ptr_fungsi)();

      ptr_fungsi=&klassx::fung_2;
      (x.*ptr_fungsi)();

      Int (klassx::*ptr_fungsi2)(int, int);
      Ptr_fungsi2=&klassx::fung_3;
     cout<<“5+8”<<(x.*ptr_fungsi2)(5,8)<<endl;
}
```