
C++
An Ever green Language
SUBRAT SWAIN - 20YRS EXPERIENCE IN C++ COACHING & DEVELOPEMENT.

Goal

100% Result Oriented Interactive Session.

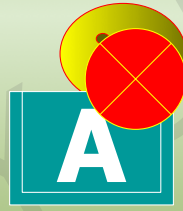
Agenda

- Revise C
- Object Oriented Analysis
- what is OOPS
- Major Pillars of Object Oriented Programming.
- What is a class & object
- Getter and Setter methods
- Constructors-Default, Parameterized
- Calling class methods
- this pointer
- static variable, static method
- constant object & constant member function
- access specifiers

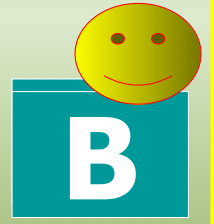

DO YOU KNOW PROGRAMMING?
(Any Language)
YES
NO

O/P?

```
main()
{
        float a=0.7;
        if(a<0.7)
            printf("A");
        else
            printf("B");
}
```




```
main()
{
        float a=5.375;
        if(a<5.375)
            printf("A");
        else
            printf("B");
}
```



Binary Of Float

What Are These?

Age Of C++!

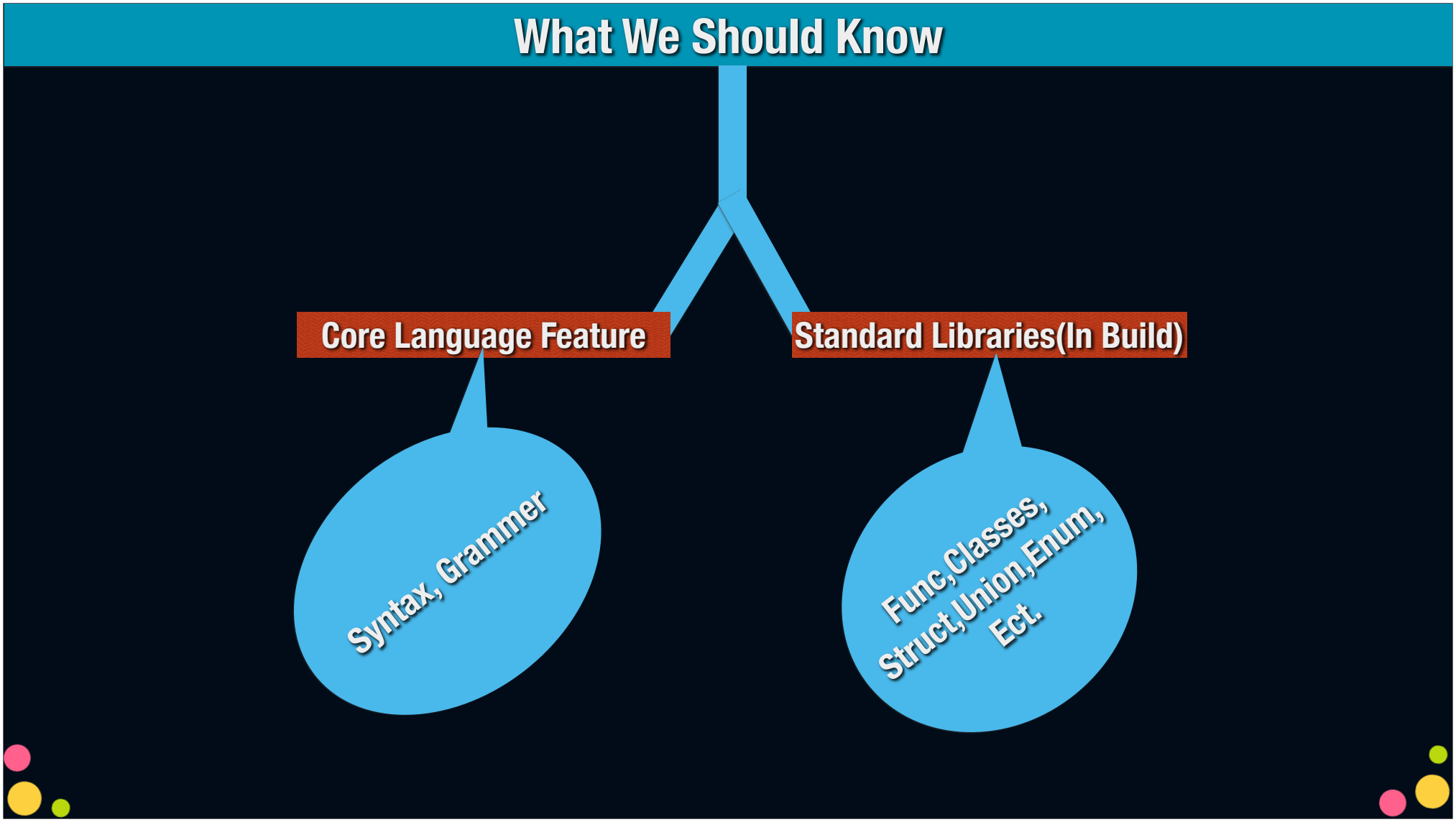
Father Of C++

Bjarne Stroustrup

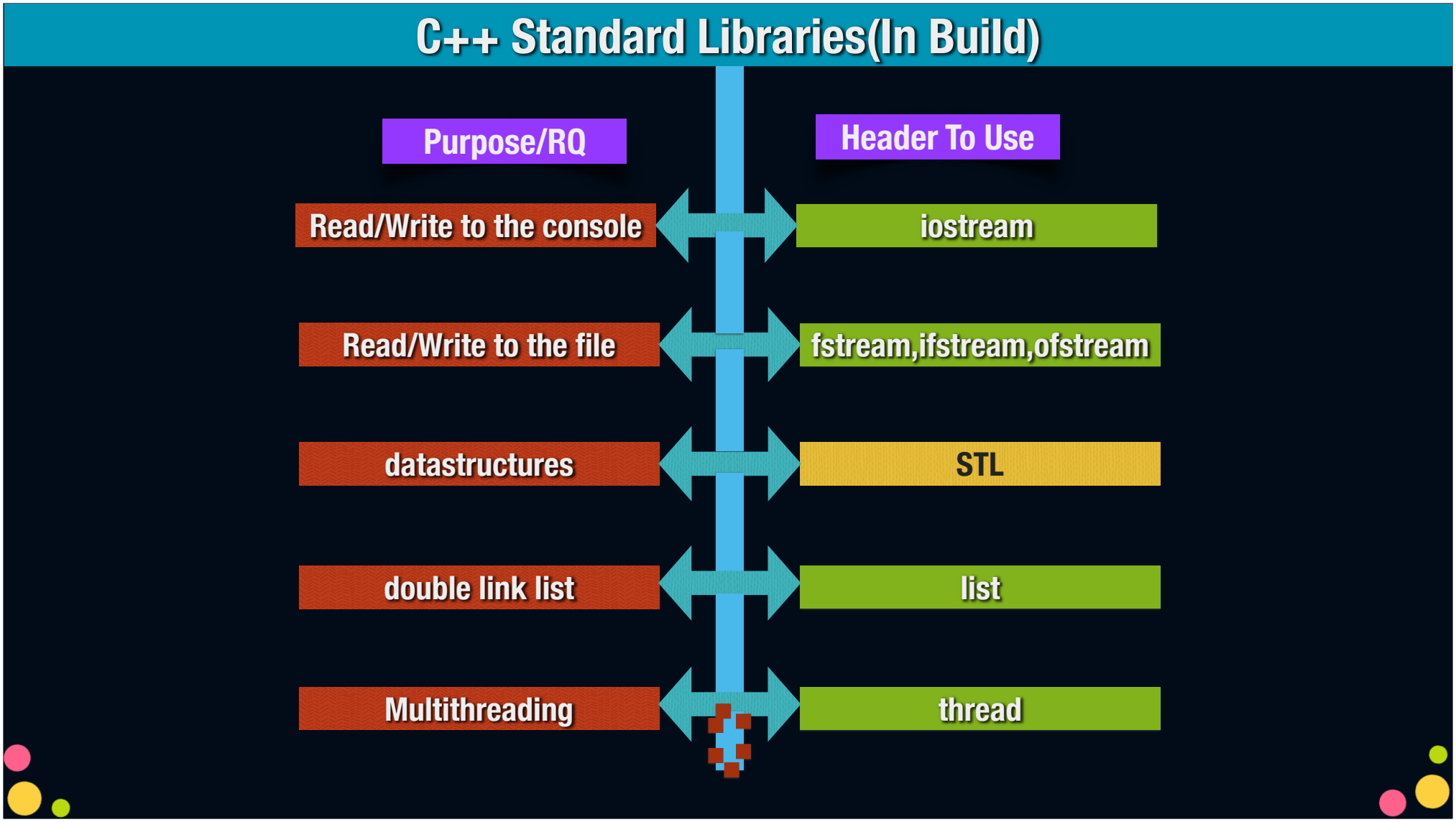
How To Master in C++ ?

1979 1998 2011 2014 2017 2020 2023 2026

MODERN C++

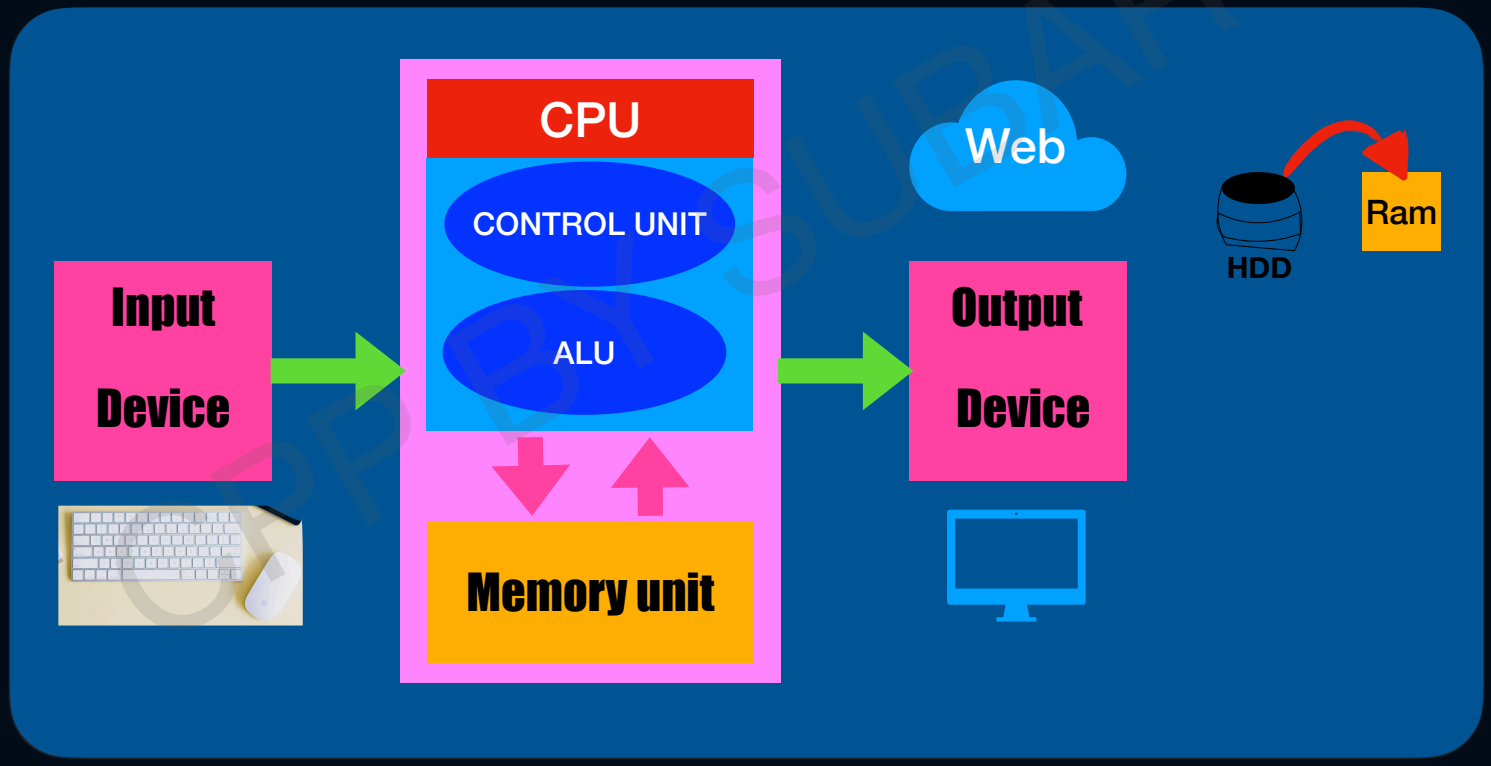

What We Should Know
Core Language Feature
Standard Libraries(In Build)
Syntax, Grammer
Func,Classes,
Struct,Union,Enum,
Ect.

WHAT IS A LIBRARY?


C++ Standard Libraries(In Build)
Purpose/RQ
Header To Use
Read/Write to the console
iostream
Read/Write to the file
fstream,ifstream,ofstream
datastructures
STL
double link list
list
Multithreading
thread

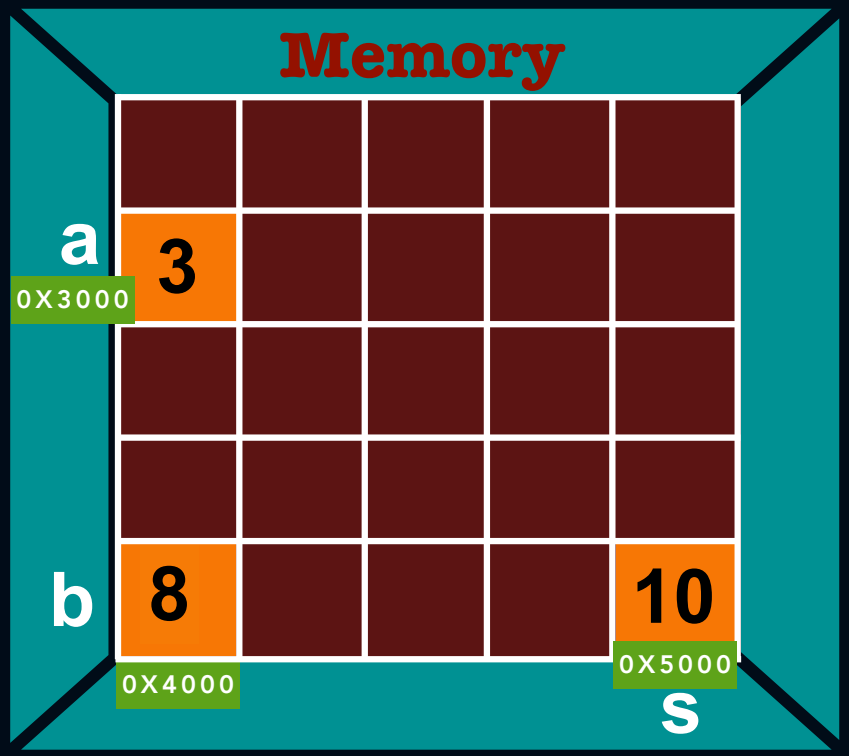
Time To Write SMALLEST
& Simplest C++ Program.

Architecture diagram Of computing system



Memory Layout…

RAM



a = 3

b = 7;

s = a + b

Print s 10

b = 8

Print b 8

Print s ~~11~~ 10

Print &b 0X4000

VARIABLE

TIPS

- To name a memory location or Slot in memory
- Value can be changed
- Has an address in memory. Use & before the variable name to get the address of a variable.

Our Strategy To Write Program

Understanding the RQ

GOING TO SUPER MARKET

WORKING WITH COMPUTER

- Store 100 — int a = 100
- Store 'A' — char ch = 'A'
- Store "Hello CPP" — string str = "HELLO CPP"
- Store 70.50 — double d = 70.50

a	ch	str	d
100	'A'	"HELLO CPP"	70.50

Contd... Working With COMPUTER

- Store 100 — int a = 100
- Store 'A' — char ch = 'A'
- Store "Hello CPP" — string str = "HELLO CPP"
- Store 70.50 — double d = 70.50

- Store n(100) no of integers
- Store n(1000) no of characters
- Store n(10000) no of doubles/floats(RealNums)

→ **Arrays**

- Store information of a person/Employee

← **Structure/Class**

TIPS

int, char, double, string => **Data Types**

Depending upon the **constant** type, we decide the **type** of variable.

NEXT C++ Program

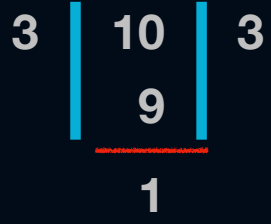
1. Write a program to print something on the screen.
2. Write a program to print current time on the screen.
3. Write a program to print the no. of cores that exist in your machine.

ASSIGNMENTS

1. Addition of Two Numbers
2. Check Even or Odd

Problem: How to check If number is even/odd?

1. Receive the number from the user `std::cin`
2. Check if the number is divisible by 2 or not
    2.1. Use if-else statement to do the check
3. If number is divisible, print “Even” otherwise print “Odd”



10/3 = 3

10%3 = 1

TIPS

- Any Number divisible by 2 is even otherwise odd
- Divisibility Test => % operator

Structures

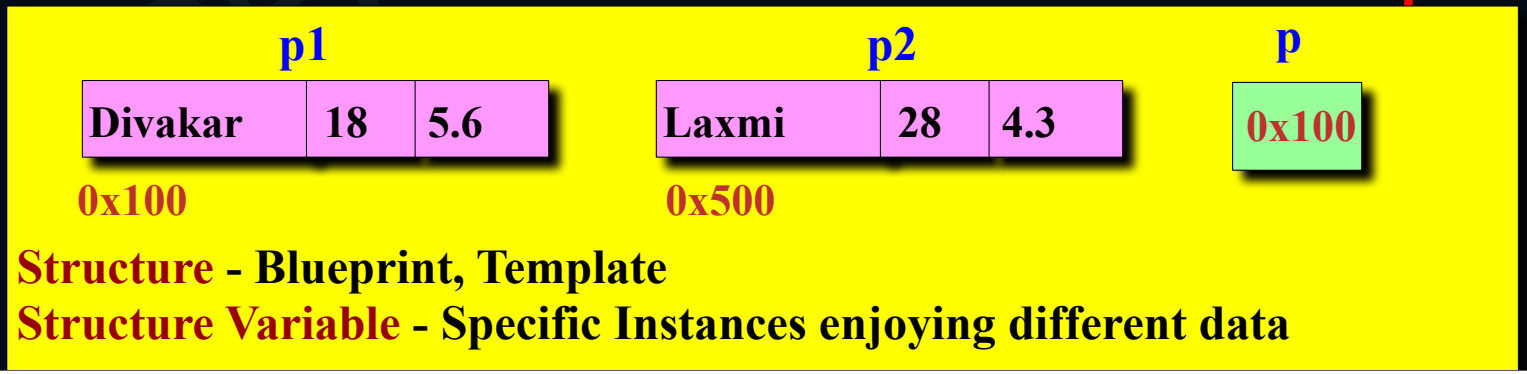
```
struct person
{
  char  n[20] ;  int  a ;  float  h ;
} ;

person  p1 = { "Divakar", 18, 5.6f } ;
person  p2 = { "Laxi", 28, 4.3f } ;

printf( "%s %d %f", p1.n, p1.a, p1.h ) ;

person*  p ;
p = &p1 ;

printf ( "%s %d %f", p->n, p->a, p->h ) ;
```

Structure Name

Structure Elements

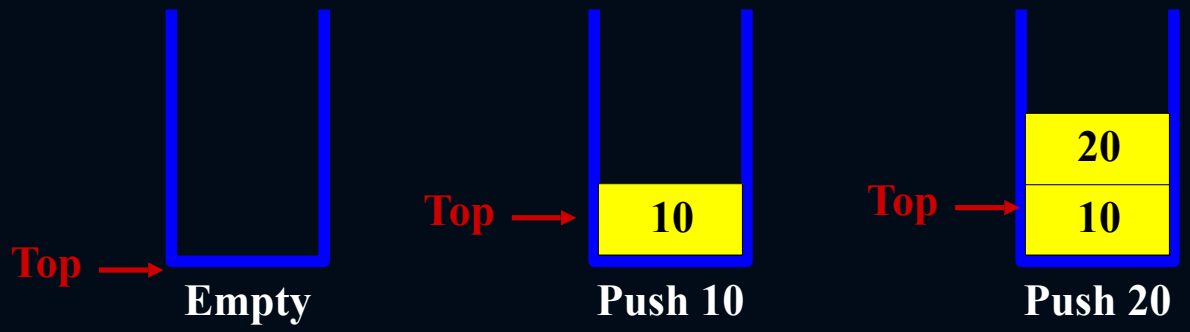
p1,p2 - Structure Var.
p - Structure Pointer
p[10] - Array of struct.

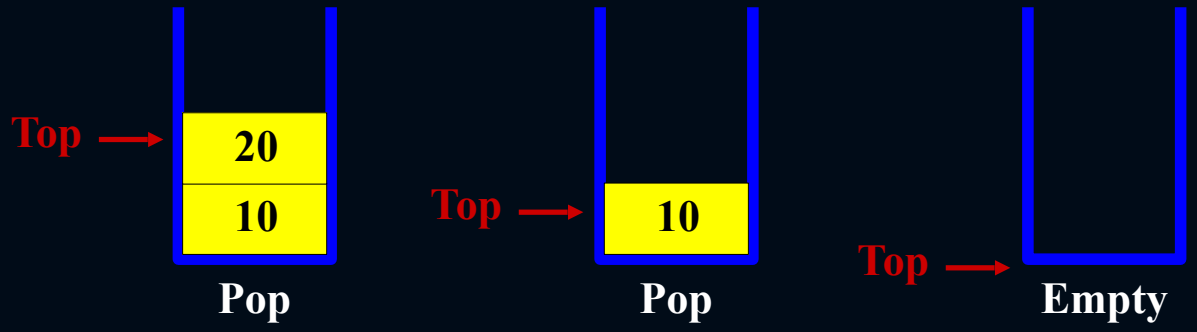
Structure Operators



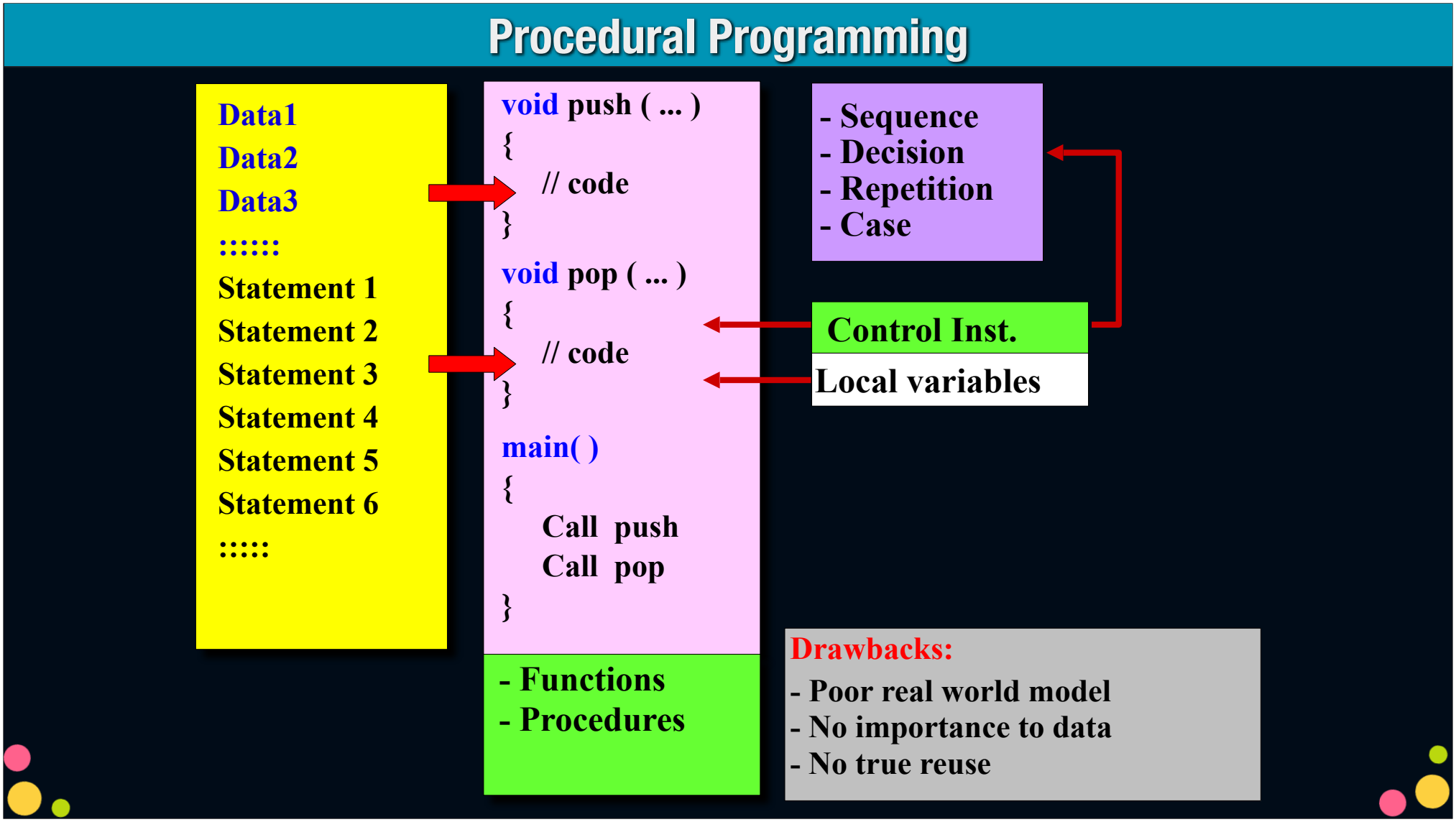
Structure - Blueprint, Template
Structure Variable - Specific Instances enjoying different data

Stack


Top
Empty
Top
10
Push 10
Top
20
10
Push 20

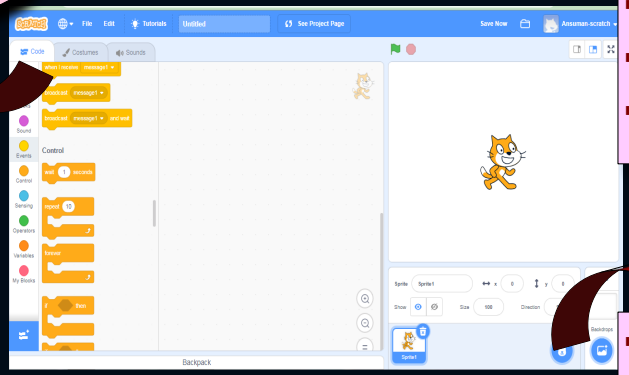

Top
20
10
Pop
Top
10
Pop
Top
Empty

Procedural Programming



Real-World Model

Objects

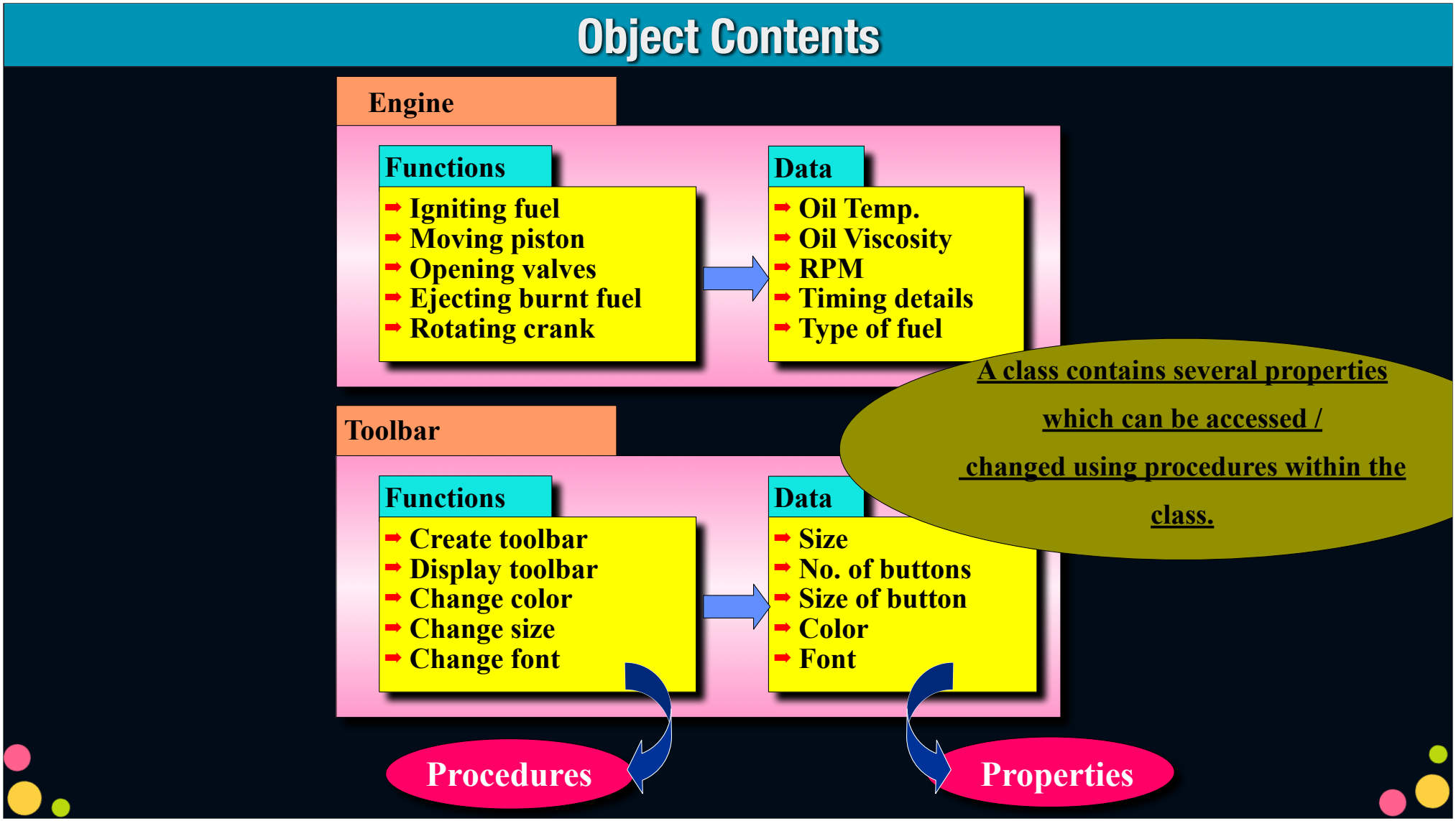


- Engine
- Gearbox
- Break
- AC
- Sound

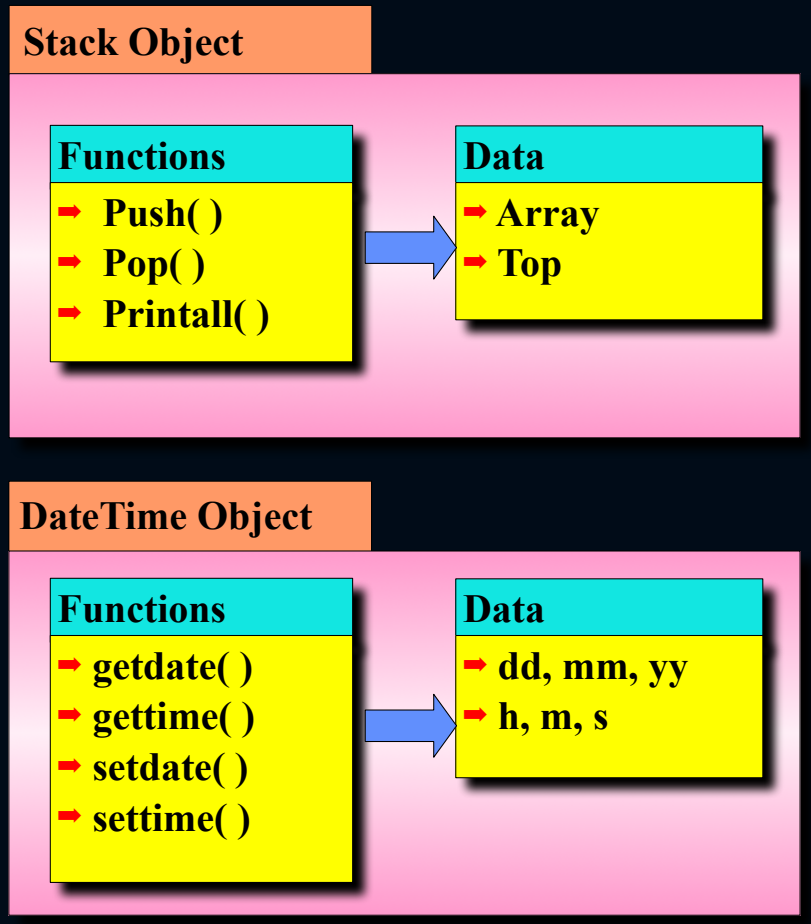
Reusable Objects

- Frame Window
- Toolbar
- Popup Menu
- Scrollbar
- Push Button

Object Contents



More Objects…



Procedural Solution

stack.c

```
struct stack
{
    int  top ;
    int  a[10] ;
} ;

void push ( ... )
{
    // code
}

void pop ( ... )
{
    // code
}
```

Use

```
struct stack s1 ;
s1.a[4] = 5 ;
s1.top = -55 ;
```

Limitations

- No importance to data
- Stress on procedures

datetime.c

```
struct  datetime
{
    int  dd, mm, yy ;
    int  h, m, s ;
} ;

 void getdate( )
 {
     // code
 }

void setdate ( ... )
{
    // code
}
```

OO Solution

stack.cpp

```
struct  stack
{
    private :
        int  top ;
        int  a[10] ;

    public :
        void push ( ... )
        {
            // code
        }

        void pop ( ... )
        {
            // code
        }
} ;
```

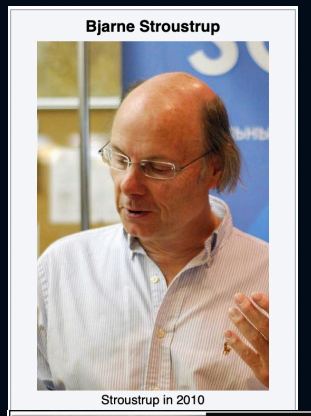
Use

```
stack  s1, s2 ;
s1.push (  .. ) ;
s1.push (  .. ) ;
s2.push ( .. ) ;
s2.push ( .. ) ;
s2.pop( ) ;
```

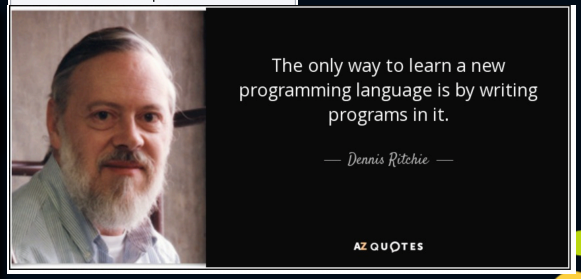
```
s1.top = -1 ;      ?
s1.a[ 4 ] = 45 ;   ?
```

Facts

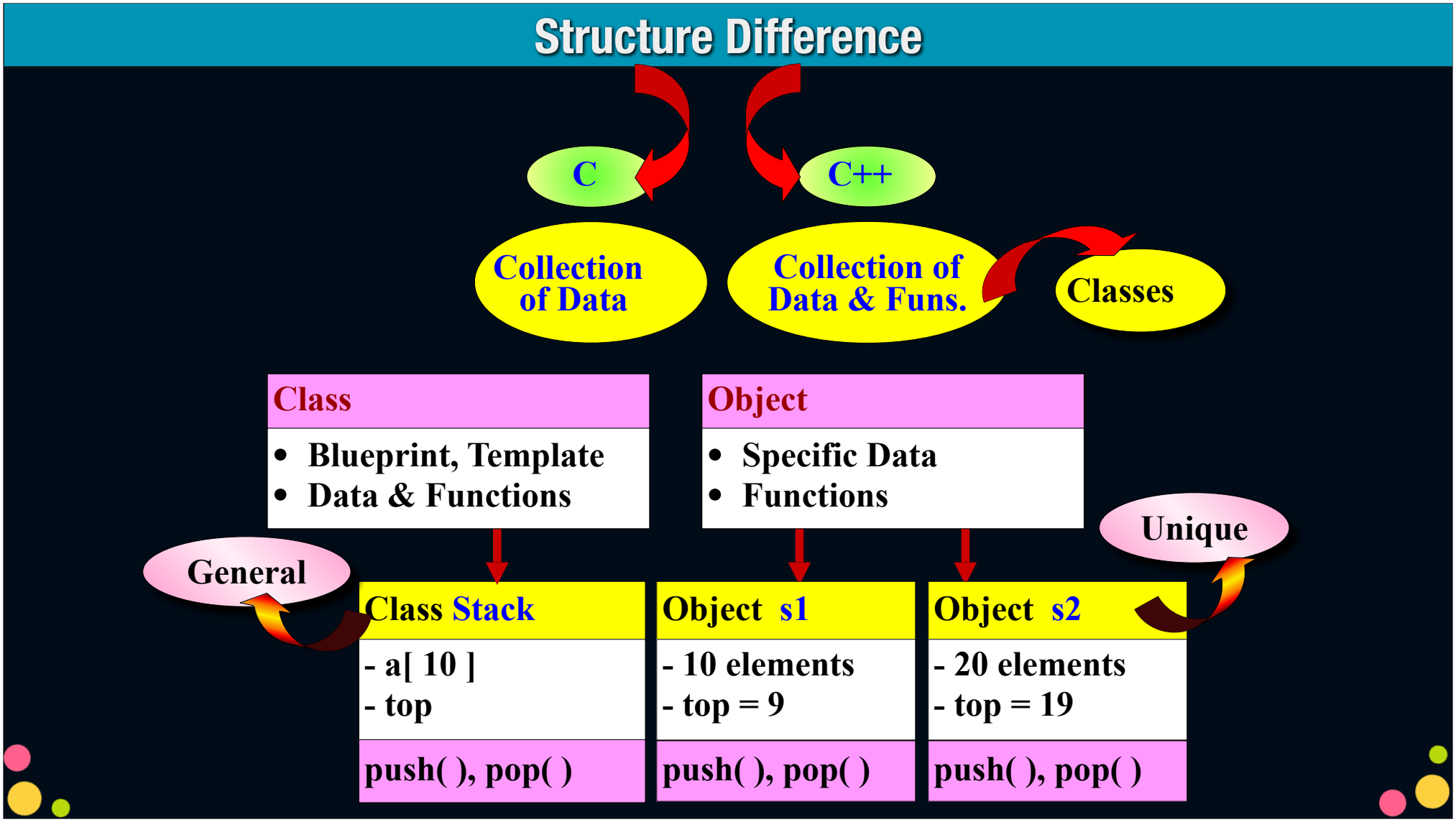
- Ex. C++, Java, C#
- 1980, AT&T Bell Labs
- Bjarne Stroustrup



Stroustrup in 2010



Structure Difference



Classes In C++

Class members are by default private

```
# include <iostream>
class  a
{
   int  i ;
   double  j ;
} ;

int main( )
{
   a  z1 = { 10, 3.14 } ;
   cout  <<  z1.i  <<  z1.j ;
   a  z2 = { 20, 6.28 } ;
   cout  <<  z2.i  <<  z2.j ;
}
```

class is optional

z1

i	j
10	3.14

z2

i	j
20	6.28

z1, z2 - Objects

cout - Console/Charecter Output
<< - Insertion Operator

Overloaded operator

```
cout  <<  z1.i  <<  z1.j ;
printf ( "%d %f", z1.i, z1.j ) ;
```

Making It Work

```
# include <iostream>
class  a
{
public :
        int  i ;  double  j ;
} ;

int main( )
{
    a  z = { 10, 3.14 } ;
    cout  <<  z.i   <<  z.j ;
}
```

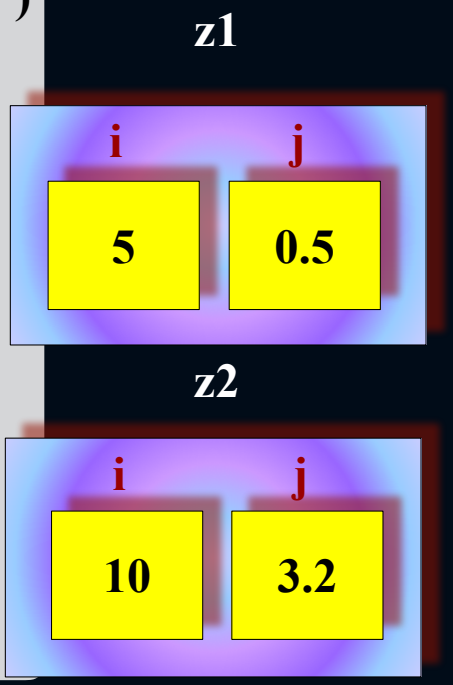
Still Better Way ..

Can functions be hidden?

Data Hiding

Encapsulation at work

```
class  a
{
   private :
      int  i ;  double  j ;

   public :

   void  setdata ( int  ii,  double  jj )
      {
            i =  ii ;  j = jj ;
      }
} ;

int main( )
{
   a   z1 , z2 ;
   z1.setdata ( 5, 0.5 ) ;
   z2.setdata ( 10, 3.2 ) ;
}
```



Which Is Better

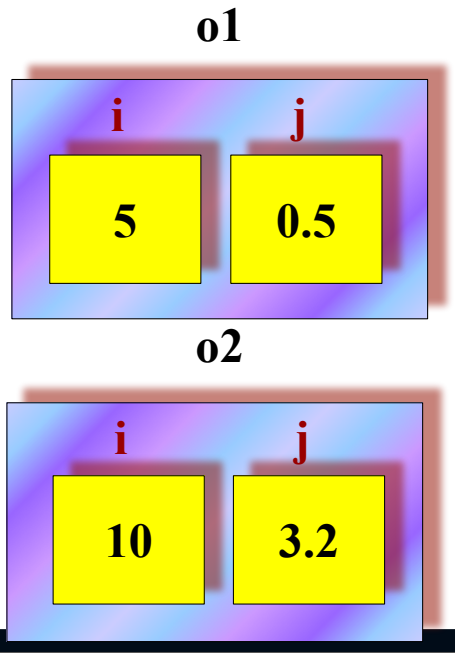
```
#include <iostream>
class person
{
    public :
        int  age ;
} ;

int main( )
{
    person  p1, p2 ; int  a ;
    cin  >>  a ;
    if ( a > 0 )
         p1.age = a ;
    cin >>  a ;
    if ( a > 0 )
        p2.age = a ;
}
```

```
#include <iostream>
class person
{
    private :
        int  a ;
    public:
        void setdata ( int x )
        {
            if ( x > 0 )
                a = x ;
        }
} ;
int main( ) {
    person  p1 ;
    person p2 ;
    int  age ;
    cin >> age ;
    p1.setdata ( age ) ;
    cin >> age ;
    p2.setdata ( age ) ;
}
```

Printing The Data

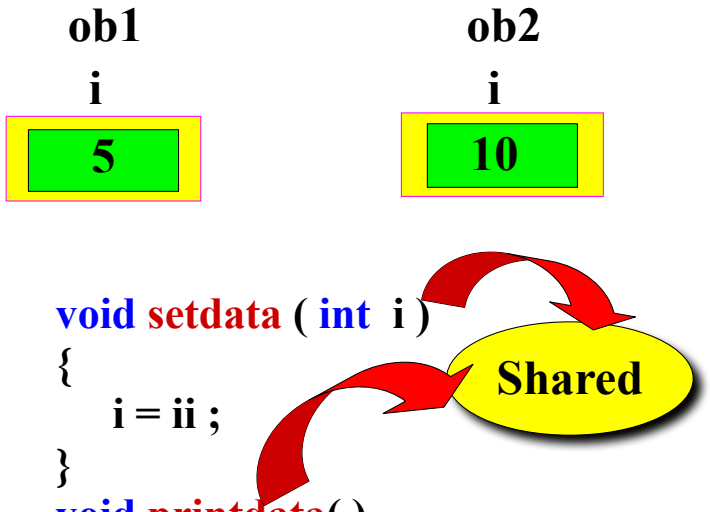
```
#include <iostream>

class a
{
   private :
      int i ; double j ;

   public :
      void setdata ( int ii, double jj )
      {
         i = ii ; j = jj ;
      }

      void printdata( )
      {
         cout << i << " " << j ;
      }

} ;
```

```
int main( )
{
    a o1 , o2 ;
    o1.setdata ( 5, 0.5 ) ;
    o2.setdata ( 10, 3.2 ) ;
    o1.printdata( ) ;
    o2.printdata( ) ;
}
```



```
class a
{
    private :
        int i ;
    public :
        void setdata ( int  ii )
        {
            i = ii ;
        }
        void printdata( )
        {
            cout << i ;
        }
} ;

int main( )
{
    a  ob1, ob2 ;
    ob1.setdata ( 5 ) ;
    ob2.setdata ( 10 ) ;
}
```

How Many Copies



Member Functions

Conceptually - Many
Practically - One

Object Oriented Analysis

https://github.com/subratswain-cpp/batch3-cpp/blob/main/ooa.md

What is OOPS

https://github.com/subratswain-cpp/batch3-cpp/blob/main/oops.md

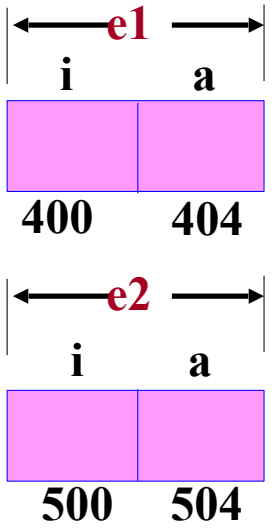
The *this* Pointer

```
class  ex
{
     private :
          int  i ;  double  a ;
     public :
          void setdata ( int  ii, double  aa )
          {
               this ->  i = ii ;
               this ->  a = aa ;
          }
} ;
int main( )
{
   ex  e1, e2 ;
   e1.setdata ( 1, 1.5 ) ;
   e2.setdata ( 2, 2.5 ) ;
}
```

```
void setdata ( ex* const this,
                    int  ii, double  aa )
```

Optional

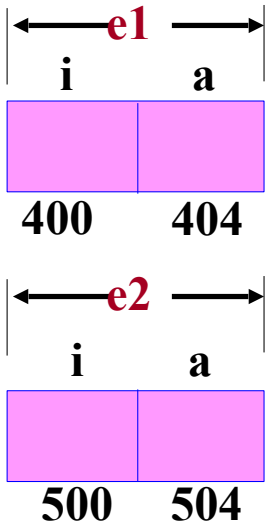
Caution!!
this Pointer cannot be modified




```
ex::setdata ( &e1, 5, 5.5 ) ;
ex::setdata ( &e2 10, 10.5 ) ;
```

Is *this* Necessary

```
class  ex
{
     private :
          int  i ;  double  a ;
     public :
          void setdata ( int  i, double  a )
          {
               this ->  i = i ;
               this ->  a = a ;
          }
} ;

int main( )
{
   ex  e1, e2 ;
    e1.setdata ( 1, 1.5 ) ;
    e2.setdata ( 2, 3.7 ) ;
}
```



Recap

- Classes = Datatypes
- Objects = Variables
- Mem Functions => Shared between objects
- This pointer => Hidden pointer to member function(NS)
- cout, cin => Objects of ostream istream class
- << is Operator ======> Function
- Access Specifiers: public, private, protected
- Can I modify private data?

BJARNE STROUSTRUP:
"THE PROTECTION OF PRIVATE CAN BE CIRCUMVENTED THROUGH POINTERS. BUT THIS, OF COURSE, IS CHEATING. C++ PROTECTS AGAINST ACCIDENT RATHER THAN DELIBERATE FRAUD."

You Try

Implement stack using the concepts that you have learned here!

C Vs. C++

```
struct stack
{
  int top;
  int a[10];
};
void init(struct stack *s)
{
     s->top=0;
}
void push(struct stack *s, int x)
{
     s->a[s->top]=x; s->top++;
}
int main()
{
  stack s1,s2;
  init(&s1);
  init(&s2);
  push(&s1,10);
  push(&s2,20);
}
```

```
class stack {
   private:
     int top;
     int a[10];
   public:
     stack()
     {
        top=0;
     }
     void push(int x)
     {
        a[top]=x;    top++;
     }
};

int main() {
     stack s1;
     s1.push(10);
     stack s2;
     s2.push(50);
}
```

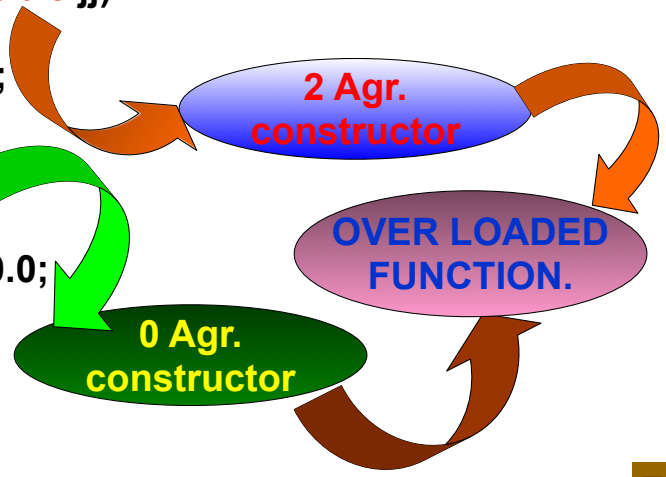
Constructor

Advantages

- ❖Cleaner
- ❖Better Organized
- ❖Sure initialisation

Constructor

```
class ex
{
 private:
    int i ;   double j;
  public :
   ex(int ii, double jj)
   {
      i = ii;j = jj;
   }
   ex( )
   {
      i = 0; j = 0.0;
   }
};

int main()
{
     ex e1(10, 20.8);

     ex e2;
}
```



e1

i	j
10	20.8

e2

i	j
0	0.0

Either I do it or you do it.

Object creation

- ➢Allocate space in memory.
- ➢Call the constructor.

Function Overloading

```
void set(int i)
{
}
void set(int i, int j)
{
}

void set(int i, double j) {

}

void set(double jj, int ii) {

}

int set(double jj, int ii) {

}
```

```
int main()
{
 set(10);
 set(10, 3);
 set(10, 3.14);
 set(3.14, 10);
 set(3.14, 10);
}
```

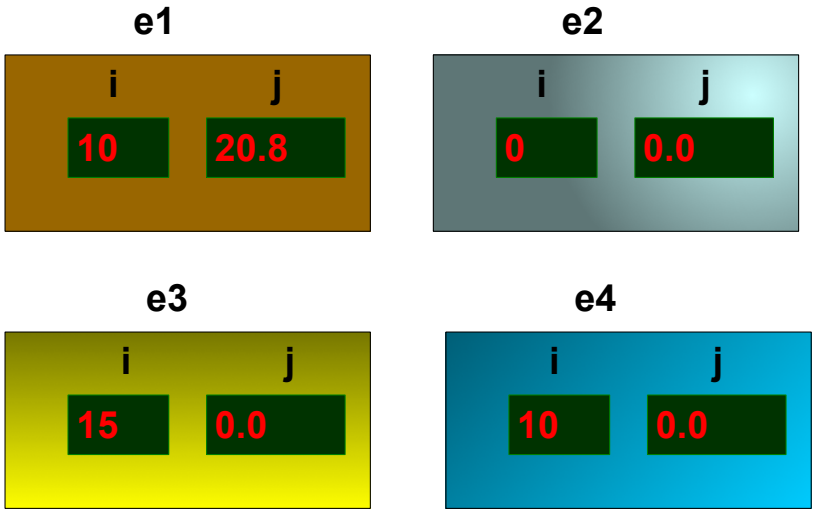
Args. Must differ in

- Number
- Order
- Type

Different return types not enough for overloading

Dual-Purpose

```
class ex
{
 private:
    int i; double j;
 public :
    ex(int ii =0, double jj= 0.0)
    {
        i = ii; j = jj;
    }
};
int main()
{
  ex e1(10,20.8);
  ex e2;
}
```



What if

```
ex e3(15)
ex e4=10;
ex e5=1, 1.1;
```

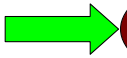
ERROR

Triple-Purpose

```
class ex
{
 private:
      int i; double f; char ch;
public :
     ex(int x = 0 , double y = 0.0, char z = 'a' )
    {
        i = x; f = y; ch=z;
    }
};
int main()
{
   ex e1(10,20.8,'o');
   ex e2(9,8.7);
   ex e3;
}
```

Trailing default Arg.

```
class ex
{
 private:
     int i; double j;
public :
     ex(int ii=0, double jj=0.0)
     {
         i = ii; j = jj;
     }
};
int main()
{
     ex e1(1);
     ex e2( ,2.9);
}
```

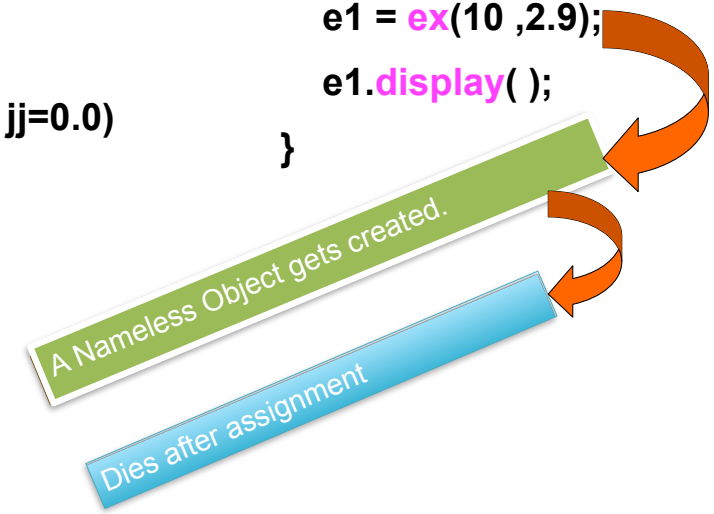


Error – prone for too Many args.

Calling Constructor Explicitly

```
class ex
{
  private:
     int i; double j;
  public :
    ex(int ii=0, double jj=0.0)
    {
         i = ii; j = jj;
    }
    void display()
    {
        cout << i << j;
    }
};
```

```
int main()
{
    ex e1;
    e1 = ex(10 ,2.9);
    e1.display( );
}
```



Array Of Objects

```
class ex
{
  private:
    int i; double j;
  public :
    ex(int ii=0, double jj=0.0)
    {
        i = ii; j = jj;
    }
    void display()
    {
        cout << i << j;
    }
};
```

- Define before use
- Doesn't die

```
int main()
{
    ex e1(1, 2.5);
    ex e2(3, 9.8);
    ex e3(4, 6.7);
    ex f[ ]={e1, e2, e3};

    ex e[ ]={
            ex(1, 2.5),
            ex(3, 9.8),
            ex(4, 6.7)
           };

  for(int i = 0; i <= 2; i++)
     e[I].display();

}
```

Pointer to an Object.

```
class ex
{
  private:
     int i; double j;
  public :
    ex(int ii=0, double jj=0.0)
    {
         i = ii; j = jj;
    }
    void set(int x, double y)
    {
      i=x; j=y;
    }
    void display()
    {
        cout << i << j;
    }
};
```

```
int main()
{
   ex e(1, 2.5);
   e.display();
   e.set(3, 5.6);
   e.display();
   fun(&e);
   e.display();
}

void fun(ex *p)
{
    p ->set(3, 5.6);
}
```

this ptr = address of e

Set new data – call set( )
Set data from other func. – pass address.

Which Gets Called

```
class ex
{
  private:
     int i; double j;
  public :
    ex(int ii=0, double jj=0.0)
    {
         i = ii; j = jj;
    }
    void set(ex obj)
    {
      i = obj.i; j = obj.j;
    }
    void display()
    {
        cout << i << j;
    }
};
```

ex obj = e1

```
int main() {
ex e1(1,2.5);
ex e2;
e2=e1;
ex e3=e1;
e2.set(e1);
ex e4=5;
e2.set(1);
e2=fun();
}
```

Calls overloaded =

Calls copy constr.

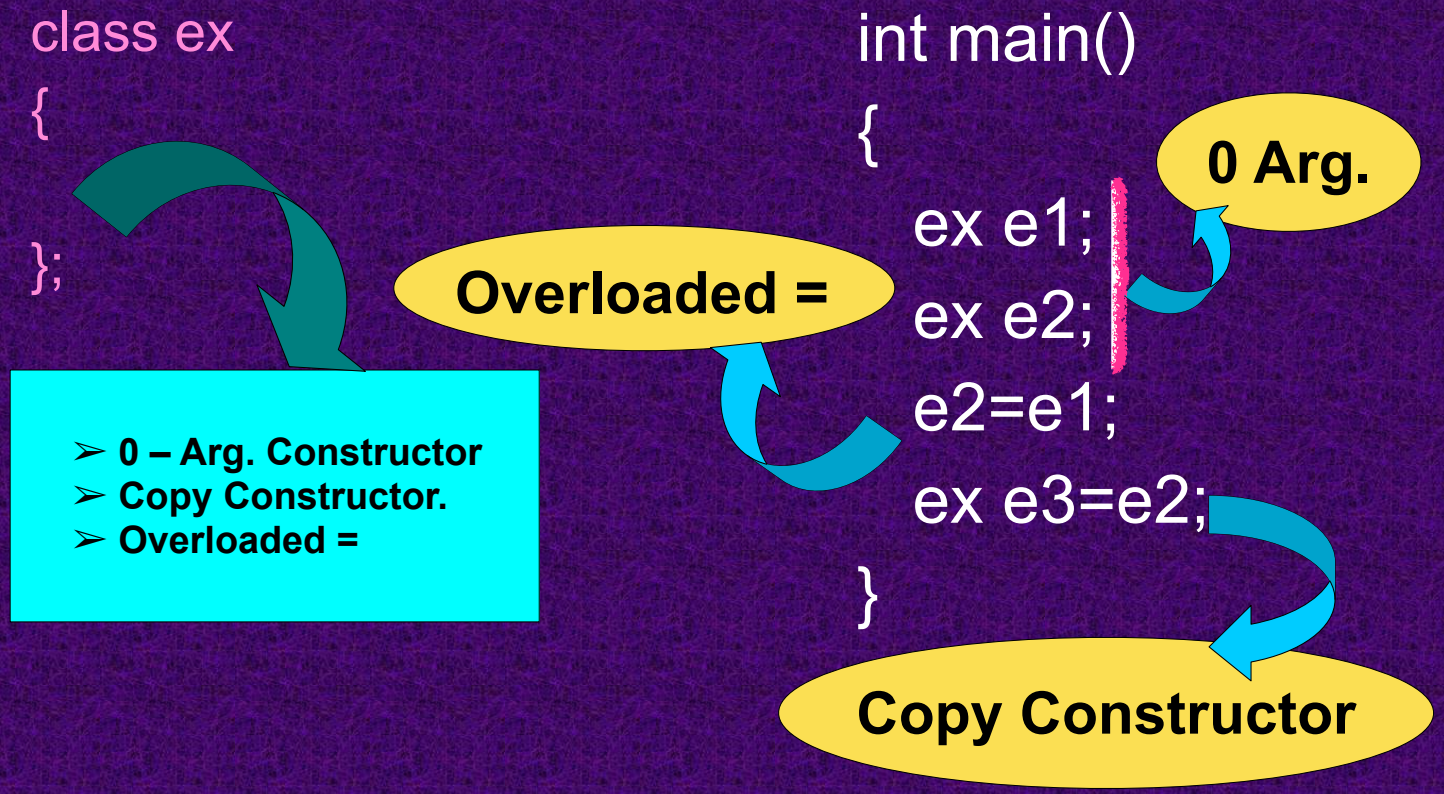
ex e4(5);

ex x(1);

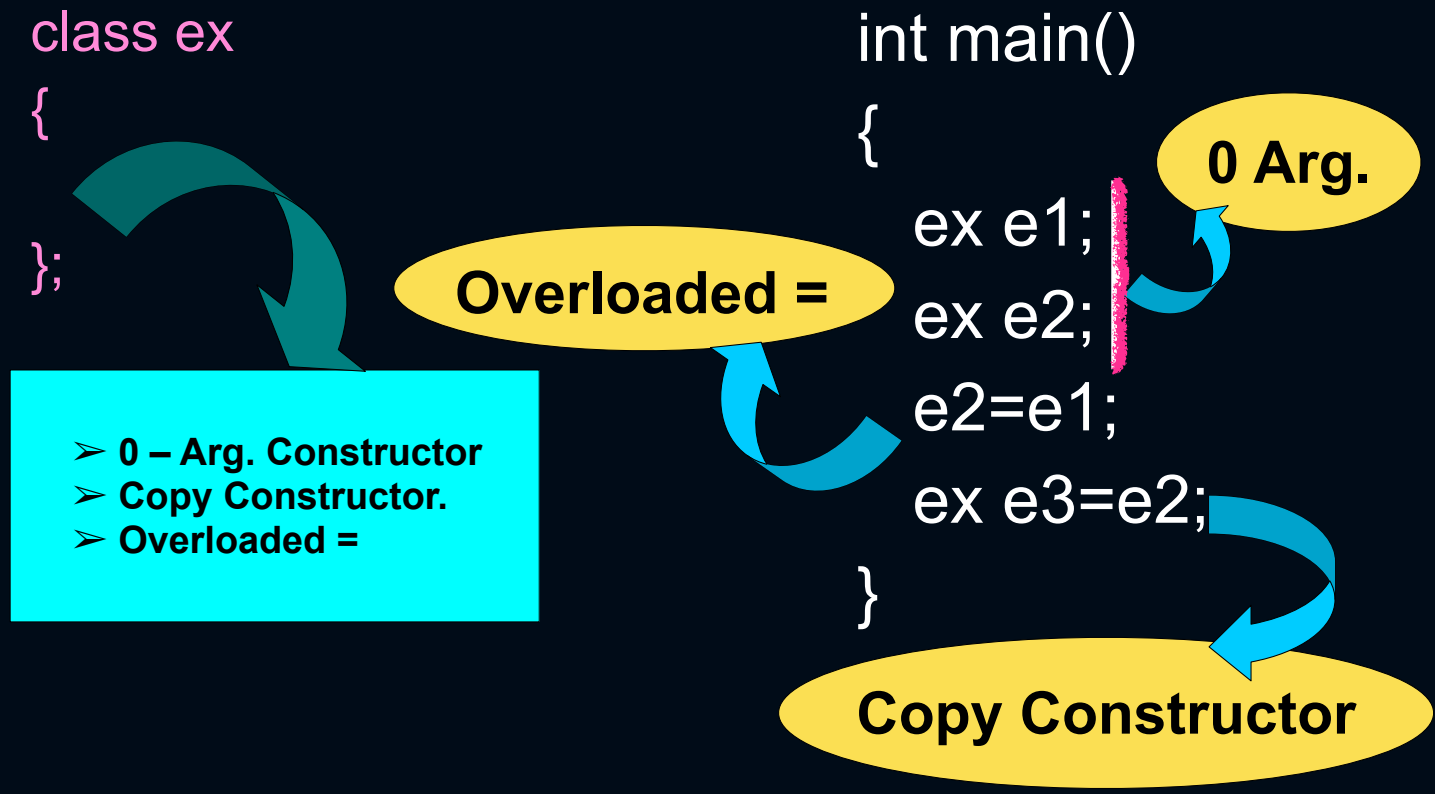
Calls copy constr. ,
Over loaded =

```
ex fun()
{
    ex t;
    return t;
}
```

Ready Mades


class ex
{
};
➢ 0 – Arg. Constructor
➢ Copy Constructor.
➢ Overloaded =
Overloaded =
int main()
{
ex e1;
ex e2;
e2=e1;
ex e3=e2;
}
0 Arg.
Copy Constructor

Ready Mades


class ex
{
};
int main()
{
ex e1;
ex e2;
e2=e1;
ex e3=e2;
}
0 Arg.
Overloaded =
Copy Constructor
➢ 0 – Arg. Constructor
➢ Copy Constructor.
➢ Overloaded =

Count No Of Objects Created From A Class

How To?

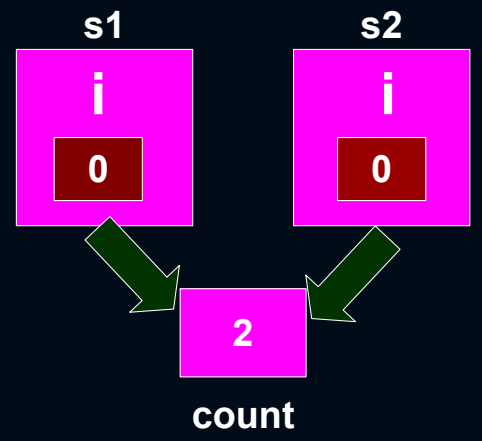
Count No Of Objects Created From A Class

Static

```
# include <iostream>
class sample
{
     int i ;
     static int count ;
   public :
     sample ( )
     {
          i = 0 ;
          count ++ ;
     }
     static void NoOfObjects( )
     {
        cout << count ;
     }
} ;

int  sample ::  count = 0 ;
```

```
s1. NoOfObjects ( ) ;
s2. NoOfObjects ( ) ;
```

```
int main ( ) {
     sample s1, s2 ;
     sample :: NoOfObjects ( ) ;
}
```



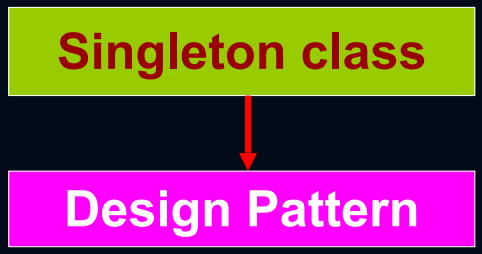
Static functions can access
Only static data

Problem

Only one object gets created from a class

Hint :

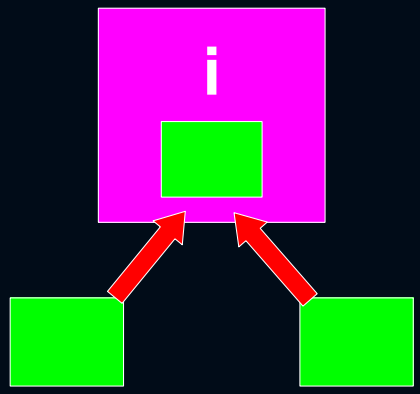
- create private constructor
- Create static member function to create object



Singleton Class

```
class sample
{
   private :
      int i ;
      static sample *p ;
      sample ( )
      {
        i = 9 ;
       }
   public :
      static sample* create ( )
      {
         if  ( p ==NULL )
               p = new sample ;
          return p ;
       }
} ;
sample*  sample:: p = NULL ;
```

```
int main ( )
{
    sample *s1 = sample ::create ( ) ;
    sample *s2 = sample:: create( ) ;
}
```



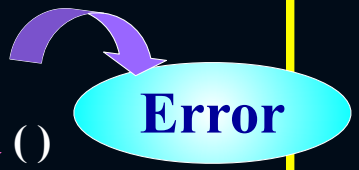
constant object & constant member function

https://github.com/subratswain-cpp/batch3-cpp/tree/main/const-correctness

Const Function

```
class sample
{
    private :
      int data ;
    public :
      sample ( )
      {
         data = 0 ;
      }
      void modifydata ( ) const
      {
         data = 20 ;
      }
      void showdata ( )
      {
          cout << data ;
      }
} ;
```

```
int  main ( )
{
     sample s1 ;
     s1.modifydata ( ) ;
     s1.showdata ( ) ;
}
```



Mutable Keyword

```
class car
{
   private :
      char model [20] ;
mutable char owner [30] ;
      int yrofmfg ;
      char regno [10] ;
   public :
      car ( char *m, char *o,
        int y, char *r )
      {
        strcpy ( model, m ) ;
        strcpy ( owner, o ) ;
        yrofmfg = y ;
        strcpy ( regno, r ) ;
   }
      void change ( char * ownr)
      {
         strcpy ( owner, ownr) ;
      }
} ;

int main ( )
{
   const car c1 ( “Camry”, ”Aansu”
        2020, “RG-H8287” ) ;
   c1.change ( “Anasuya” ) ;
}
```

access specifiers

Any Questions?

Thank You

Book 1: Object Oriented Programming with C++ 8th Edition by E Balagurusamy.

- Online reference:

https://cppreference.com

https://www.nextptr.com/

- Additional E- BooK:

E-book: C++ How to Program 10th Edition by Paul Deitel and Harvey Deitel

Subrat Swain

DAY-2

Session 2 :
Introduction to Object Oriented Programming & Classes and Objects destructor, memory leakage, dangling pointer, copy constructor, ~~function overloading, default value argument to a function.~~

ASSIGNMENTS

1. WAP to impelemt a banking system.

Agenda

- Dynamic Memory Allocation
- Static V/s Dynamic
- New Vs Malloc
- Memory Leak
- How to avoid ML
- Array Allocation
- Destructor
- Dangling pointer
- Copy constructor
- Function overloading
- Default value argument to a function

Dynamic Memory Allocation

```
int i ;
float a ;
struct emp
{
    char n[ 20 ] ;
    int id ;
    float s ;
} ;
emp e ;
```

New

-Operator
-Does DMA

```
int *i ;     float *a ;
i = ( int * )  malloc ( sizeof (int ) ) ;
a = ( float *) malloc ( sizeof (float) ) ;
struct emp
{
      char n[20] ;
      int id ;
      float s ;
} ;
emp *e ;
e = (emp * )  malloc ( sizeof ( emp ) ) ;
```

i = new int ;

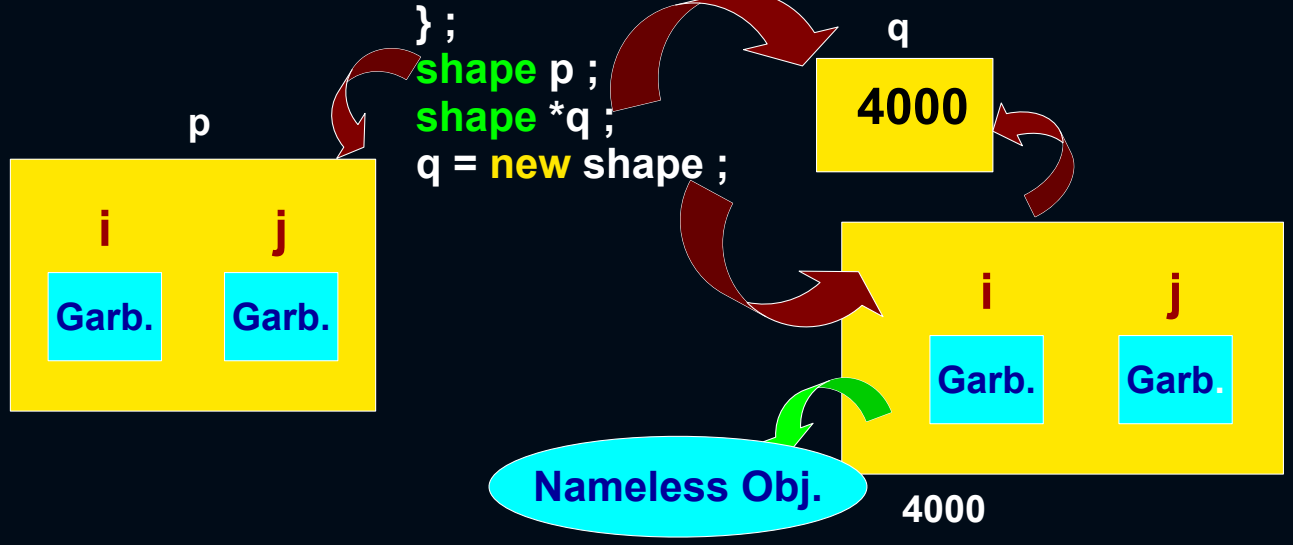
a = new float ;

e = new emp ;

Static V/s Dynamic

What's The Difference

```
class shape
{
    private:
        int i, j ;
} ;
shape p ;
shape *q ;
q = new shape ;
```



Are *new* And *Malloc* ( ) Same

```
class ex
{
   private :
        int i ; double a ;
   public :
        ex ( )
        {
          i = 0 ;
          a = 0.0 ;
         }
         ex ( int ii, double aa )
         {
             i= ii ;
             a= aa ;
          }
} ;
```

```
int main ( )
{
    ex *p1 = new ex ;
    ex *p2 = new ex ( 10, 3.5 ) ;
    delete p1 ;
    delete p2 ;
}
```

Operator

- New allocates memory, calls constructor
- What is allocated must be de-allocated
- Delete calls destructor, de-allocates memory

Avoid Memory Leaks - I

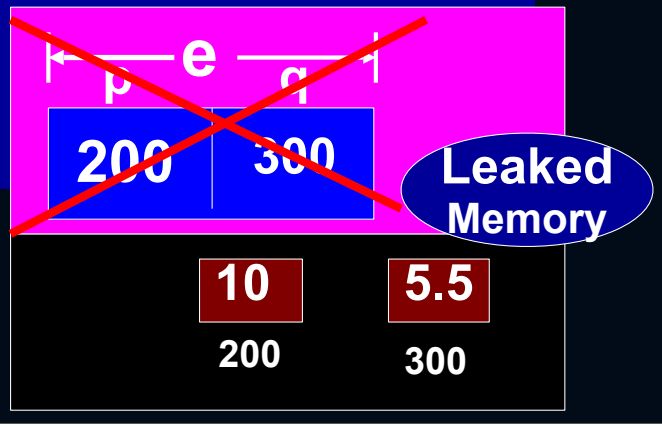
```
class ex
{
   private :
      int *p ; double *q ;
   public :
      ex ( int ii, double aa )
      {
          p = new int ;
          q = new double ;
          *p = ii ;
          *q = aa ;
       }
       ~ ex ( )
       {
          delete p ;
          delete q ;
       }
} ;
```

Destructor

Doesn't delete q

```
int main ( )
{
    void f( ) ;
    f ( ) ;
}
void f ( )
{
    ex  e ( 10, 5.5) ;
}
```

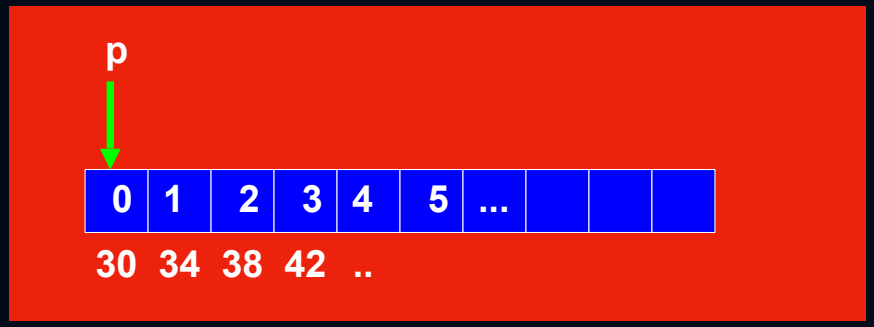
If new is used in constructor, use delete in destructor



Array Allocation

```
int *q ;
q = new int ;

int *p ;
p = new int [10] ;
for ( int i = 0 ; i < 10 ; i++ )
        * ( p + i ) =  i ;
```

Can be a variable

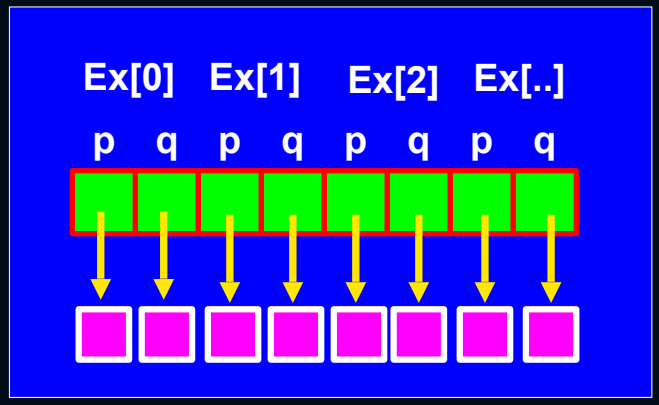


Avoid Memory Leaks - II

```
class ex
{
    private :
      int *p ; double *q ;
    public :
      ex ( )
      {
          p = new int ;
          q = new double ;
          *p = 0 ;
           *q = 0 ;
       }
       ~ex ( )
      {
          delete p ;
          delete q ;
      }
} ;
```

```
int main ( )
{
        void f( ) ;
         f ( ) ;
}
void f ( )
{
        ex *z ;
         z = new ex[ 10 ] ;
         delete z ;
}
```

delete [ ] z ;



dangling pointer

copy constructor

https://github.com/subratswain-cpp/batch3-cpp/tree/main/copy-constructor

https://learn.microsoft.com/en-us/cpp/cpp/constructors-cpp?view=msvc-170#copy_and_move_constructors

https://www.simplilearn.com/tutorials/cpp-tutorial/cpp-copy-constructor

Any Questions?

THANK YOU!