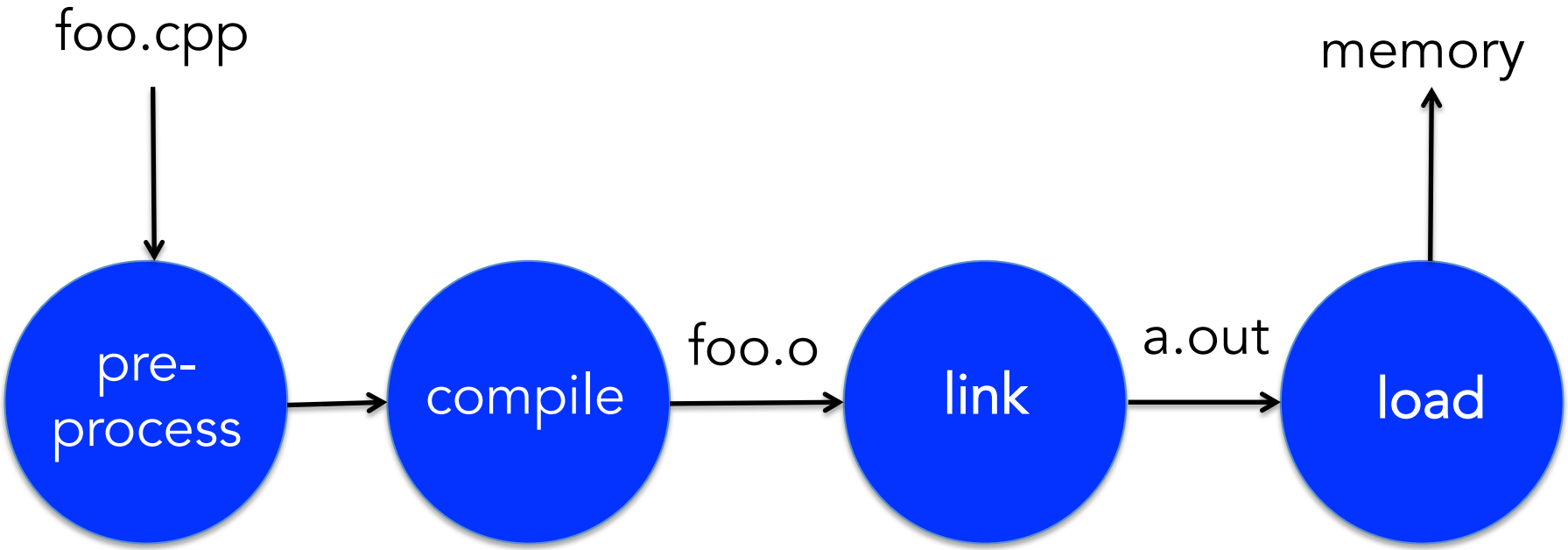
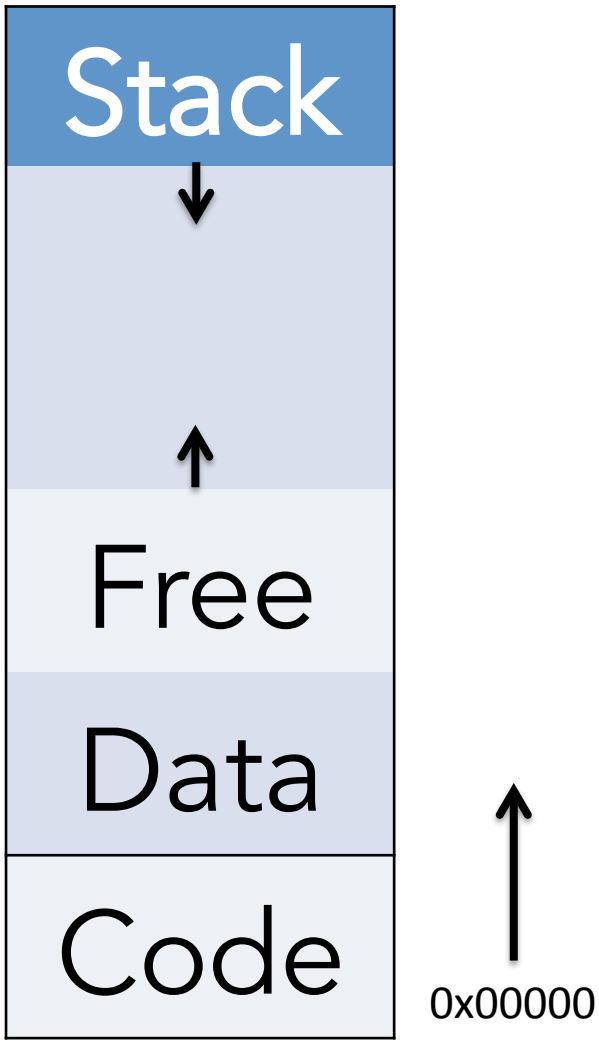
Lecture 1
C++ Object Oriented Programming

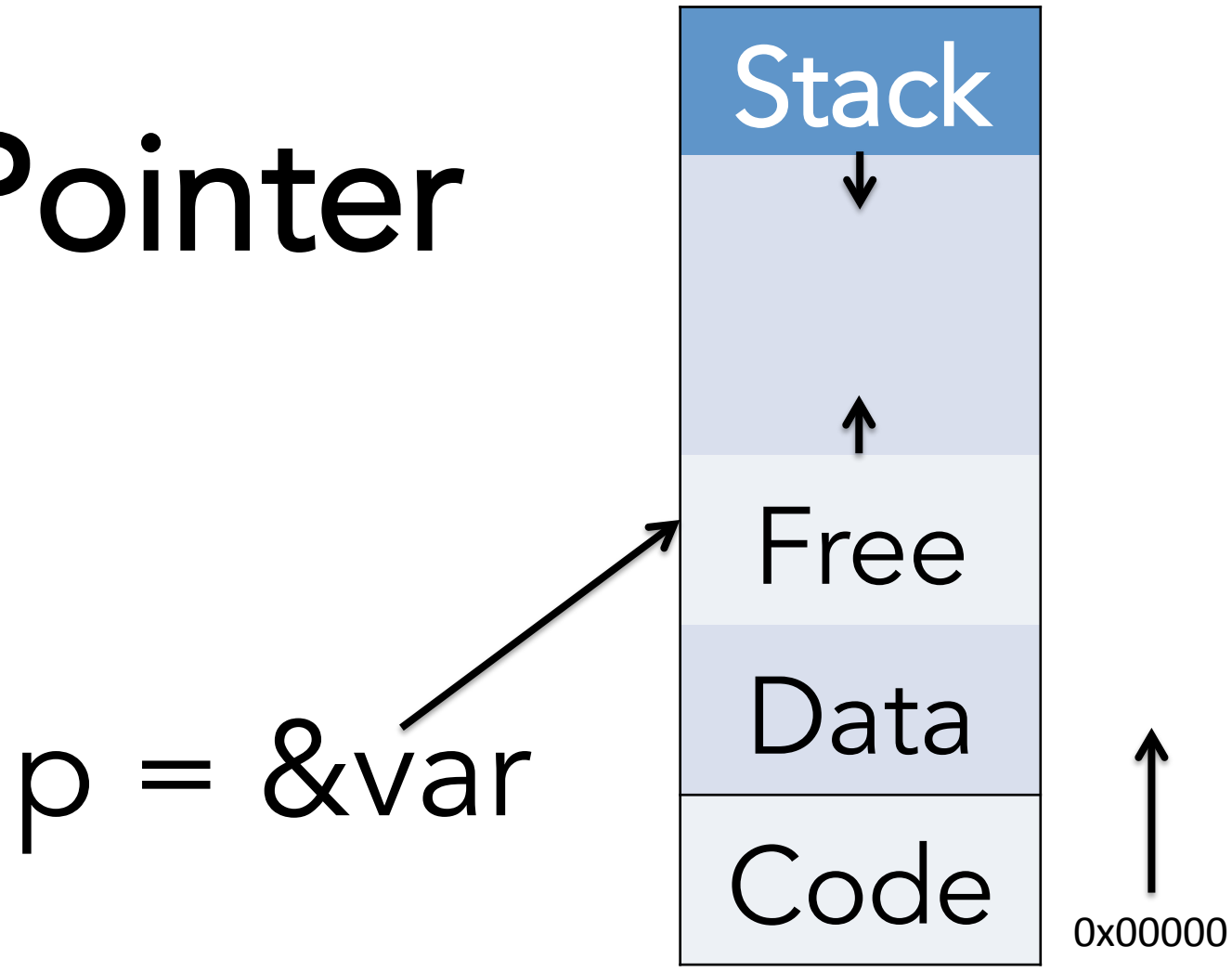
Why are we studying C++ in a data structures course?


foo.cpp
pre-
process
compile
foo.o
link
a.out
load
memory

Program Memory Layout



Pointer



with a pointer we:

- can access address

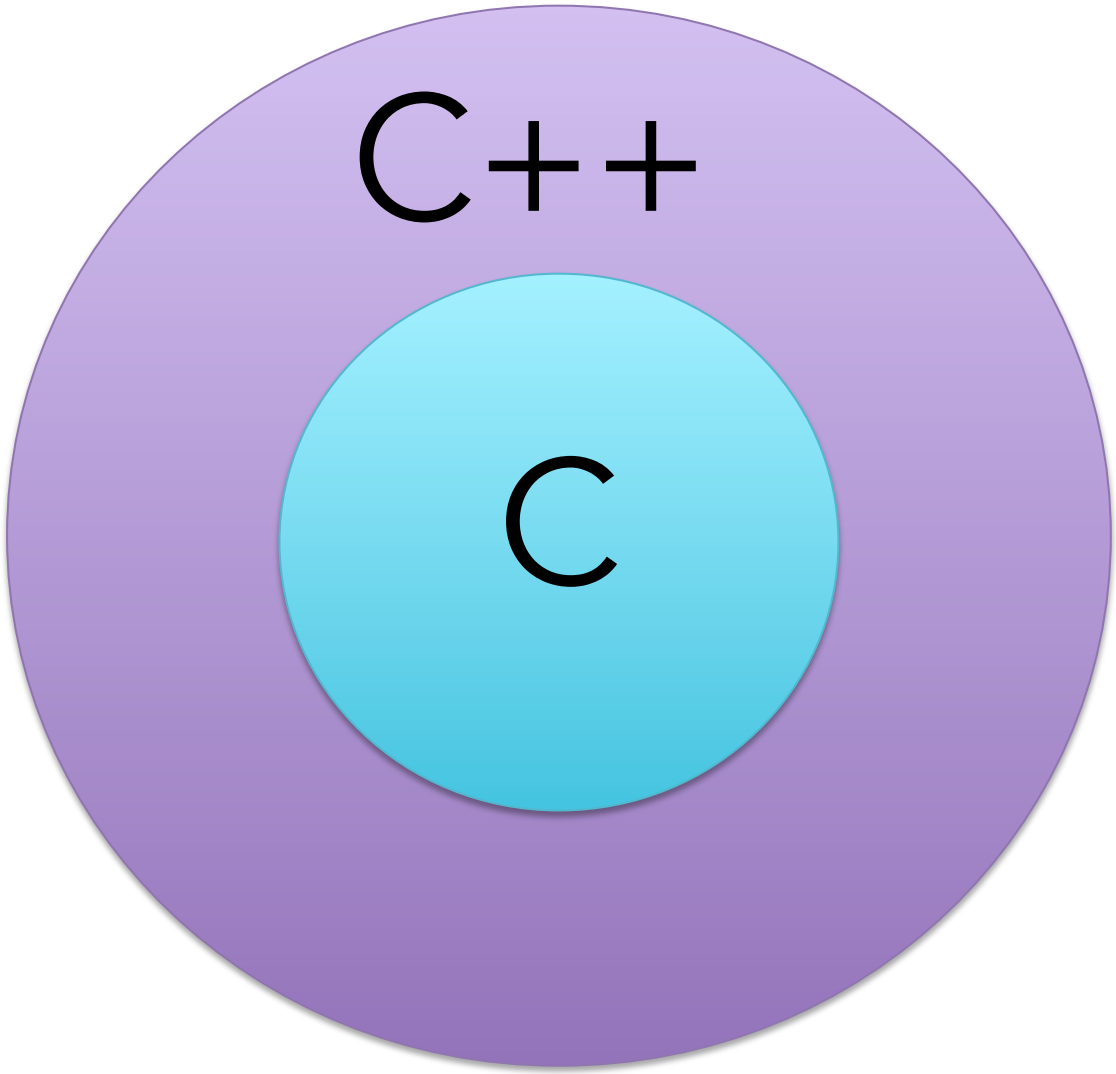
address=p

- can access content

content=*p

How are pointers relevant to data structures?

What happens when you compare two pointers?


C++
C

New in C++

bye bye printf, scanf
welcome cin, cout

```
cin>>x>>y;
cout<<x<<"And"<<y;
```

malloc -> new

free -> delete

New data type bool = {true, false}

```
bool isEven(int x){
    if (x%2 == 0)
        return true;
    else
        return false;
}
```

Inline comments

```
float prices[100]; // Array storing prices of books
```

instead of only /* Comment*/ in C

Declare Anywhere

C

```
int main(){
    int i,j,k;
    i=6;
    j=4;
    k=i+j;
    return k;
}
```

C++

```
int main(){
    int i,j;
    i=6;
    j=4;
    int k=i+j;
    return k;
}
```

Finding maximum in C

```
int maxInt(int A[], int n){
    cmax=A[0];
    for (int i=1;i++;i<n)
        if (A[i]>cmax)
            cmax = A[i];
    return cmax;
}
```

```
float maxFlt(float A[], int n){
    cmax=A[0];
    for (int i=1;i++;i<n)
        if (A[i]>cmax)
            cmax = A[i];
    return cmax;
}
```

Finding maximum in C++

```
int max(int A[], int n){
    cmax=A[0];
    for (int i=1;i++;i<n)
        if (A[i]>cmax)
            cmax = A[i];
    return cmax;
}
```

```
float max(float A[], int n){
    cmax=A[0];
    for (int i=1;i++;i<n)
        if (A[i]>cmax)
            cmax = A[i];
    return cmax;
}
```

Function overloading

Even better, you can have input types as arguments!

```
int X = {2, 4, 7, 7 ,9, 3, 6};
double Y = {0.3, 0.55, 0.2, 0.7, 0.5};

int maxX = max<int>(X, 7);
double maxY = max<double>(Y, 5);
```

Function templates

```
template<typename T> T max(T A[], int n)
      cmax=A[0];
     for (int i=1;i++;i<n)
          if (A[i]>cmax)
               cmax = A[i];
     return cmax;
}
```

Default Arguments

```
//Declaration
double log(double a, int b=2);

//Usage
log(100,10);//Means log10100

log(double 100);//Means log2100
```

How would you compare two structures?

```
struct dob {
int year;
int month;
int day; };
```

```
struct dob x;
struct dob y;
//some code that assigns values to x and y
if(x.day==y.day &&
x.month==y.month &&
x.year==y.year)
//do something
```

```
bool operator == (struct dob x, struct dob y) {
    if(x.day==y.day && x.month==y.month && x.year==y.year)
        return true;
    else
        return false;
}
//Usage
if (x==y)
    //do something
```

Operator Overloading

Function Inlining

The called function is literally expanded in the calling function!

In C++, strings can be concatenated with "+"!

```
str1 = "Mukesh";
str2 = "Saini";
str3 = str1 + "Kumar" + str2;
//now str3 is Mukesh Kumar Saini
```

C++ has Classes and Objects!

IEEE Spectrum

The Top Programming Languages 2016

C Structures

```
struct Passenger {
    string name;
    dob age;
    int air_miles;
    struct flights[];
    struct routes;
};
```

```
Passenger Ram;
//Some code
flight L =longestFlight(Ram.flights);

route E =mostEconomic(Ram.routes);
```

We want function and data together!

C Structures

```
struct Passenger {
    string name;
    int age;
    int air_miles;
    struct flights[];
    struct routes;
};
```

We want protection from outside code!

C Structures

```
struct Passenger {
    string name;
    int age;
    int air_miles;
    struct flights[];
    struct routes;
};
```

We want easy re-use of the code!

Class Examples

```
class students{
    …
    private int i,j,k;
    public int COMP(i,j){
        //Compare rank
    }
  …
}
```

```
class books{
    …
    private int i,j,k;
    public int COMP(i,j){
        //Compare price
    }
  …
}
```

Data Abstraction

Only essential information is provided to the outside world, details are hidden!

Encapsulation

Data and functions are kept in a safe unit called Object!

Inheritance facilitates code reuse!

```
class MTech_Students: public students{
    private float gate_score;
    ...
    ...
}
```

Encapsulation
Abstraction
Inheritance

Class variables are called objects!

Object Examples

```
Students st_ref = new Students();
```

```
Books bk_ref = new Books();
```

Class Examples

```
class students{
    …
    private int i,j,k;
    public int COMP(i,j){
        //Compare rank
    }
  …
}
```

```
class books{
    …
    private int i,j,k;
    public int COMP(i,j){
        //Compare price
    }
  …
}
```

Why use Classes for Data Structures?

Data Types

1. Primitive
2. Composite
3. Abstract

Primitive Data Types

- bool
- short
- int
- long
- char
- float
- double
- enum

Composite Data Types

Arrays

Abstract Data Types

- Stack – push/pop
- List – add to front, visit elements
- Queue – add to back, remove from front
- Trees
- and more . . .