

Announcements

- **Regrades of scheme project out soon (grading criteria already on website)**
- **C project due in one week (April 5)**
- **This week's office hours will be in ARC; times will be extended; there will be some "last minute" time too on Monday evening**
- **Make sure your remus account works (if it didn't last time!)**

1

C++

- **Language basics - comparison to Java**
- **Operator overloading**
- **Pointers versus references**
- **Dynamic dispatch**
- **Visibility**

2

C++

- **An object-oriented language built on C in mid-1980's**
- **Multiple inheritance**
 - **Operator overloading - use of existing operators with new types or arguments**
- **Objects: on a stack frame versus in the heap**
- **Use of pointers and reference variables**
 - **Use pointers to refer to dynamically created objects, not references**
- **Choice of direct or dynamic dispatch (virtual functions)**

3

C++ versus Java

- **Pointers to objects**
- **Multiple inheritance**
- **Objects can be created statically or dynamically**
- **Virtual functions dynamically bound**
- **Operator overloading**
- **Class implementation can be defined separately from class specification**
- **C syntax**
- **Allows global procedure definition**

- **References to objects (more restricted than pointers)**
- **Single inheritance + interfaces**
- **All objects created dynamically**
- **All functions dynamically bound**
- **No operator overloading**
- **Class implementation with class specification**
- **C-like syntax**
- **All procedures and functions associated with a class**

4

C++ references

References are the type for aliases (marked by a `&`) that:

- do not need explicit dereferencing (in fact, there is no way of getting at its memory in any other way)
- must be initialized (can't be reassigned later, and can't be NULL)

```
int i = 3;
int &j = i;    (compare to int *p = &i;)
j++;
    ← i is now 4
```

5

C++ references

- Often used for call-by-reference

```
void swap(int &a, int &b)
{
    int tmp=a; a=b; b=tmp;
}
...
int x = 3, y = 2;
swap(x, y);
    ← x is 2, y is 3
```

6

C++ references

- or returned by a function

```
int &index(int *arr, int i)
{
    return arr[i];
}
```

...

```
int a[4] = { 10, 20, 30, 40 };
// Like saying a[2]=3
index(a,2) = 3;
```

7

Constructor calling example

```
class X
{
public:
    X() { printf("Constructor called\n"); }
    ~X() { printf("Destructor called\n"); }
    void f(int i=0) { cout << "i is " << i << "\n"; }
} x2;
main()
{
    printf("main started\n");
    X x1;
    x1.f();
}
```

Output

```
Constructor called
main started
Constructor called
i is 0
Destructor called
Destructor called
```

8

Subtyping and Derived Classes

- A derived class inherits some behavior from its base class
`class V : public A`
- A subtype value can be used anywhere its supertype value can be used.
- If a public derived class inherits all members from its base class, without overriding any, then it has a subtyping relation with its base class.

9

C++ Class Example

```
#include <stdio.h>
#include <stream.h>
class vector
{
    int sz;
    int *v;
public:
    vector(int);    constructor
    ~vector() {delete v;}    destructor
    int size() const {return sz;}    member
    int& elem(int i) const {return v[i];}    functions
    int& operator[](int) const;
};
```

C++ allows overloading of operators:
e.g., subscript operator []

10

C++ Class Example, cont.

```
void error(char *s)    procedure
{ cout << s << "\n";
  exit(1);
}
vector::vector(int i)  constructor implementation
{ if (i <= 0) error("bad vector size");
  sz = i;
  v = new int[i];
}
int& vector::operator[](int i) const
{ if (i < 0 || i >= sz) error("index out of bounds");
  return v[i];
}
```

**overloaded subscript operator code
does bounds check so is safe! elem()
is not safe.**

11