2/19/2014

Pointers vs. References

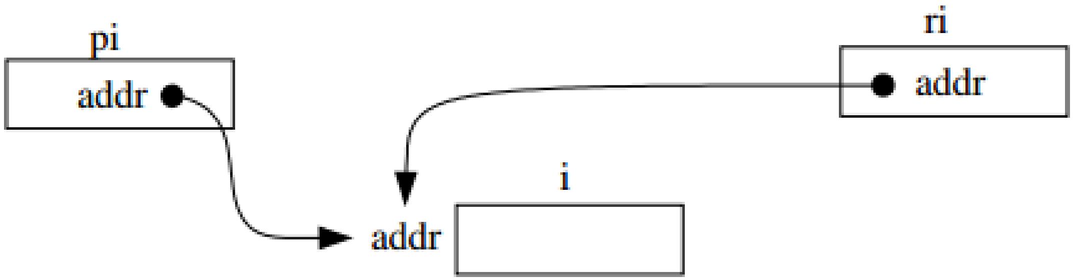
Moritz Petry

GERMAN
CANCER RESEARCH CENTER
IN THE HELMHOLTZ ASSOCIATION

50 Years – Research for
A Life Without Cancer

Basics

Pointer & reference hold the address of the object.



Basic differences

Pointer

Does not have to be initialized:

```
int i = 0;

int * pi;    // Okay. But risky.
pi = &i;
```

Uses pointer arithmetic:

```
pi++;      // Increment pointer.
```

Can point to NULL:

```
int * pi = 0;
```

Reference

Has to be initialized:

```
int i = 0;
int & ri;   // Error, reference has to
               be initialized.
int &ri = i;
```

Same arithmetic as referred object:

```
ri++;      // Increment i.
```

One chance to refere to NULL:

```
int * pi = 0;
int & ri = *pi;
```

Cannot refer to NULL!

Benefits of references

Pointer

Need to be dereferenced:

```
*pi = 5;
```

Longer syntax:

```
vector<int> v(10);
vector<int>* pv = &v;
pv->size();
```

Check whether pointer is NULL:

```
void printP (const int * p)
  {
    if (p)
      cout << * p;
  }
```

Reference

No operators needed:

```
ri = 5;
```

Same syntax as referred Object:

```
vector<int> v(10);
vector<int>& rv = v;
rv.size();
```

No need to check:

```
void printR (const int & r)
  {
    cout << r;
  }
```

Operators

Pointer

Not usual arithmetic layout:

```
MyVector* operator +(const MyVector & i);
MyVector* operator =(const MyVector & i);
```

```
(*v4) = (*v1) + (*v2) + (*v3);
```

Looks like a vector of pointers:

```
MyVector* operator [](const int & i);
```

```
*v[1] = 15;
```

Reference

Familliar arithmetic layout:

```
MyVector& operator +(const MyVector & i);
MyVector& operator =(const MyVector & i);
```

```
v4 = v1 + v2 + v3;
```

Normal vector syntax:

```
MyVector& operator [](const int & i);
```

```
v[1] = 15;
```

Conclusion

- If you know you have something to refere to, use a reference.
- If you do not want to reassign your pointer, use a reference.
- If you create your own operators, use almost every time a reference.
- Use pointers in all other cases.

2/19/2014

Thank you for your attention!