FUNCTION RETURN, PART 2

Returning structured data

Delroy A. Brinkerhoff

EXAMPLE DATA

```
struct part
{
        char   type;
        int    id;
};
```

RETURN BY VALUE

```
part supplier()
{
      part  a = { 'd', 10 };

      return a;
}
```

```
void client()
{
      part x = supplier();
}
```

RETURN BY POINTER: LOGICAL ERROR

```
part* supplier()
{
        part  a = { 'd', 10 };

        return &a;
}
```

```
void client()
{
        part* x = supplier();
}
```

RETURN BY POINTER: CORRECT OPTIONS

STATIC DATA

```
part* supplier()
{
        static part  a = { 'd', 10 };

        return &a;
}
```

DYNAMIC DATA

```
part* supplier()
{
        part*  a = new part;
        a->type = 'd';
        a->id = 10;

        return a;
}
```

RETURN BY REFERENCE: LOGICAL ERROR

```
part& supplier()
{
        part  a = { 'd', 10 };

        return a;
}
```

```
void client()
{
        part& x = supplier();
}
```

RETURN BY POINTER: CORRECT OPTIONS

STATIC DATA

```
part& supplier()
{
        static part   a = { 'd', 10 };

        return a;
}
```

DYNAMIC DATA

```
part& supplier()
{
        part*   a = new part;
        a->type = 'd';
        a->id = 10;

        return *a;
}
```

SPECIAL CONSIDERATIONS

STATIC DATA

- If the function reads or calculates new data on each call, the data from the previous call is overwritten
- Therefore, previous data must be fully processed before calling the function again

DYNAMIC DATA

- Memory allocated with the new operator must be deallocated with the delete operator

RETURNING NON-LOCAL DATA: RETURN BY POINTER

```
part* supplier(part* a)
{
        a->type = 'd';
        a->id = 10;

        return a;
}
```

```
void client()
{
        part y;

        part* x = supplier(&y);
}
```

RETURNING NON-LOCAL DATA: RETURN BY REFERENCE

```
part& supplier(part& a)
{
        a.type = 'd';
        a.id = 10;

        return a;
}
```

```
void client()
{
        part y;

        part& x = supplier(y);
}
```

RETURN BY REFERENCE FUNCTIONS FORM AN L-VALUE

```
int main()
{
    part p;
    part r;

    r.type = 'x';
    r.id = 50;
    supplier(p) = r;

    cout << p.type << " "
        << p.id << endl;

    return 0;
}
```