

## Contents

<b>1</b>	<b>Data Types and Variables</b>	<b>2</b>
1.1	Types . . . . .	2
1.2	Declaration vs Definition . . . . .	2
<b>2</b>	<b>Polymorphism</b>	<b>3</b>

# 1 Data Types and Variables

## 1.1 Types

A selection of data types that always seem to be on the exam:

```
int a
    An integer.

int *a
    A pointer to an integer.

char a[3]
    An array of characters of length 3.

int *a[3]
    An array of pointers to integers of length 3.

int (*a)[3]
    A pointer to an array of integers of length 3.

int char *(*a)[]
    A pointer to an array of pointers to characters.

const int * a
    A pointer to a const integer.

int * const a
    A const pointer to an integer
```

## 1.2 Declaration vs Definition

### Declaration

- Provides basic attributes: type and name
- Does not allocate storage (for variables)
- e.g. `extern int a;`

### Definition

- Defines all attributes of a symbol
- e.g. `int a;`

A symbol being declared but never defined results in a linker error.

## 2 Polymorphism

---

```
1 class ITouhou {
2     public:
3     ITouhou() { cout << "ITouhou (" << name() << ") constructed" << endl; }
4     virtual ~ITouhou() { cout << "ITouhou destructed" << endl; }
5     virtual string name() const { return "ITouhou"; };
6 };
7
8 class Youkai : public virtual ITouhou {
9     public:
10    Youkai() : ITouhou() { cout << "Youkai (" << name() << ") constructed" << endl; }
11    virtual ~Youkai() { cout << "Youkai destructed" << endl; }
12    virtual string name() const { return "Youkai"; };
13 };
14
15 class IFlyable: public virtual ITouhou {
16     public:
17     IFlyable() { cout << "IFlyable (" << name() <<") constructed" << endl; }
18     virtual ~IFlyable() { cout << "IFlyable destructed" << endl; }
19     virtual void fly() { cout << "IFlyable::fly()" << endl; }
20     virtual string name() const { return "IFlyable"; }
21 };
22
23 class Yuuka : public Youkai, public IFlyable {
24     public:
25     Yuuka() : Youkai(), IFlyable()
26     { cout << "Yuuka " << name() << " constructed" << endl; }
27     virtual ~Yuuka() { cout << "Yuuka destructed" << endl; }
28     virtual string name() const { return "Yuuka"; }
29     virtual void fly() { cout << "Yuuka::fly()" << endl; }
30 };
31
32 class Reimu : public virtual ITouhou, public IFlyable
33 {
34     public:
35     Reimu() : ITouhou(), IFlyable()
```

---

Listing 1: Polymorphism example classes

```
1   virtual ~Reimu() { cout << "Reimu destructed" << endl; }
2   virtual string name() const { return "Reimu"; }
3 };
4
5 int main() {
6     ITouhou *y1 = new Yuuka();
7     delete y1;
8
9     cout << "=====" << endl;
10
11    Youkai *y2 = new Yuuka();
12    delete y2;
13
14    cout << "=====" << endl;
15
16    Yuuka *y3 = new Yuuka();
17    y3->fly();
18    delete y3;
19
20    cout << "=====" << endl;
21
22    IFlyable *y4 = new Yuuka();
23    y4->fly();
24    delete y4;
25
26    cout << "=====" << endl;
27
28    IFlyable *r1 = new Reimu();
29    r1->fly();
30    delete r1;
31
32    return 0;
33 }
```

---

Listing 2: Polymorphism example main()

---

```
1 ITouhou (ITouhou) constructed
2 Youkai (Youkai) constructed
3 IFlyable (IFlyable) constructed
4 Yuuka Yuuka constructed
5 Yuuka destructed
6 IFlyable destructed
7 Youkai destructed
8 ITouhou destructed
9 =====
10 ITouhou (ITouhou) constructed
11 Youkai (Youkai) constructed
12 IFlyable (IFlyable) constructed
13 Yuuka Yuuka constructed
14 Yuuka destructed
15 IFlyable destructed
16 Youkai destructed
17 ITouhou destructed
18 =====
19 ITouhou (ITouhou) constructed
20 Youkai (Youkai) constructed
21 IFlyable (IFlyable) constructed
22 Yuuka Yuuka constructed
23 Yuuka::fly()
24 Yuuka destructed
25 IFlyable destructed
26 Youkai destructed
27 ITouhou destructed
28 =====
29 ITouhou (ITouhou) constructed
30 Youkai (Youkai) constructed
31 IFlyable (IFlyable) constructed
32 Yuuka Yuuka constructed
33 Yuuka::fly()
34 Yuuka destructed
35 IFlyable destructed
36 Youkai destructed
37 ITouhou destructed
38 =====
39 ITouhou (ITouhou) constructed
40 IFlyable (IFlyable) constructed
41 Reimu (Reimu) constructed
42 IFlyable::fly()
43 Reimu destructed
44 IFlyable destructed
45 ITouhou destructed
```

---

Listing 3: Polymorphism example output

Notes:

- The closest implementation of a `virtual` function to the type of the instance will be called
- The implementation of a function not declared `virtual` will be that of the type (not the type if the instance)
- For this reason destructors should always be `virtual`
- Virtual inheritance is used to avoid the "diamond pattern" problem when a class inherits from multiple children of a single base class
- Without virtual inheritance this results in multiple copies of the base class being created