(1)

#include<iostream>

using namespace std;

void binaryOut(char c) {

unsigned char tmp = 0x80;

for (int i = 0; i<8; i++) {

if (c&tmp)

cout << 1;

else

cout << 0;

tmp >>= 1;

}

}

void displayInBinary(void \*data, int len) {

for (int i = len - 1; i >= 0; i--) {

binaryOut(\*((char\*)data + i));

}

cout << endl;

}

int main() {

char c = 'b';

short n = 257;

float f = 1.0f;

displayInBinary(&c, sizeof(c));

displayInBinary(&n, sizeof(n));

displayInBinary(&f, sizeof(f));

system("pause");

return 0;

}

(2)

Source-code file:

#include<iostream>

using namespace std;

class shape {

public:

void draw() {

cout << "can not draw a general shape!" << endl;

}

};

class triangle :public shape {

public:

void draw() {

cout << " \* " << endl;

cout << " \* \* " << endl;

cout << "\*\*\*\*\*" << endl;

}

};

class rectangle :public shape {

public:

void draw() {

cout << "\*\*\*\*\*" << endl;

cout << "\* \*" << endl;

cout << "\*\*\*\*\*" << endl;

}

};

int main(int argc, char \*argv[]) {

shape c; triangle t; rectangle r; shape \*p;

shape &rt = t; shape &rr = r;

c.draw(); t.draw(); r.draw();

p = &c;

p->draw(); p = &t; p->draw();

p = &r;

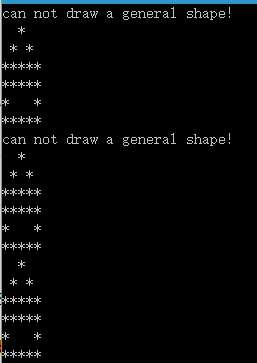
p->draw(); rt.draw(); rr.draw();

system("pause");

return 0;

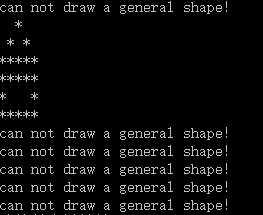
}

Q1:



Yes.

Q2:



Q3:

The usage of virtual function is to achieve polymorphism. Different kind of objects can call the function to achieve different work by using virtual. Also, when a pointer is pointed to a polymorphism class object, complier can call the corresponding function which is virtual.