**НАСЛЕДЯВАНЕ**

Наследяване – механизъм, по който един клас може да наследи свойствата на друг.

Базов клас /base/– наследяваният клас, чиито свойства се наследяват

Производен клас /derived/ - класът, който наследява основните характеристики на базовият клас и добавя свои собствени.

1. class B { //дефинира базов клас
2. int i;
3. public:
4. void set\_i(int n);
5. int get\_i();
6. };
7. class D : public B { //дефинира производен клас
8. int j;
9. public:
10. void set\_j(int n);
11. int mul();
12. };

Пример 1: Това е програма, използваща класовете B и D

1. #include <iostream>
2. using namespace std;
3. class base { //дефинира базов клас
4. int i;
5. public:
6. void set\_i(int n);
7. int get\_i();
8. };
9. class derived : public base { //дефинира производен клас
10. int j;
11. public:
12. void set\_j(int n);
13. int mul();
14. };
15. void base::set\_i(int n) //задаване на стойности на i в класа base
16. {
17. i = n;
18. }
19. int base::get\_i() //връщане на стойност на i в класа base
20. {
21. return i;
22. }
23. void derived::set\_j(int n) //задаване на стойност на j в класа derived
24. {
25. j = n;
26. }
27. int derived::mul() //връщане на стойността на i от base умнож. с j
28. {
29. //производният клас може да извиква public член-функции на базовия клас
30. return j \* get\_i();
31. }
32. int main()
33. {
34. derived ob;
35. ob.set\_i(10); // зареждане на i в base
36. ob.set\_j(4); // зареждане на j в derived
37. cout << ob.mul(); // извежда 40
38. return 0;
39. }

Пример 2. Тази програма дефинира базов клас, който описва определени характеристики на плод. Класът е наследен от два производни класа Apple и Orange, които добавят специфични характеристики за съответните класове.

1. #include <iostream>
2. #include <cstring>
3. using namespace std;
4. enum yn {no, yes};
5. enum color {red, yellow, green, orange};
6. void out(enum yn x);
7. char \*c[] = { "red", "yellow", "green", "orange"};
8. class fruit { //общ клас за плод fruit
9. public:
10. enum yn annual;
11. enum yn perennial;
12. enum yn tree;
13. enum yn tropical;
14. enum color clr;
15. char name[40];
16. };
17. class Apple : public fruit { // производен клас Apple
18. enum yn cooking;
19. enum yn crunchy;
20. enum yn eating;
21. public:
22. void seta(char \*n, enum color c, enum yn ck, enum yn crchy,
    1. enum yn e);
23. void show();
24. };
25. class Orange : public fruit { //производен клас orange
26. enum yn juice;
27. enum yn sour;
28. enum yn eating;
29. public:
30. void seto(char \*n, enum color c, enum yn j, enum yn sr,
    1. enum yn e);
31. void show();
32. };
33. void Apple::seta(char \*n, enum color c, enum yn ck,
    * 1. enum yn crchy, enum yn e)
34. {
35. strcpy(name, n);
36. annual = no;
37. perennial = yes;
38. tree = yes;
39. tropical = no;
40. clr = c;
41. cooking = ck;
42. crunchy = crchy;
43. eating = e;
44. }
45. void Orange::seto(char \*n, enum color c, enum yn j,
    * 1. enum yn sr, enum yn e)
46. {
47. strcpy(name, n);
48. annual = no;
49. perennial = yes;
50. tree = yes;
51. tropical = yes;
52. clr = c;
53. juice = j;
54. sour = sr;
55. eating = e;
56. }
57. void Apple::show()
58. {
59. cout << name << " apple is: " << endl;
60. cout << "Annual: "; out(annual); //едногодишно растение
61. cout << "Perennial: "; out(perennial); //многогодишно растение
62. cout << "Tree: "; out(tree); //расте на дърво
63. cout << "Tropical: "; out(tropical); //тропично
64. cout << "Color: " << c[clr] << endl; //има цвят
65. cout << "Good for cooking: "; out(cooking); //става за готвене
66. cout << "Crunchy: "; out(crunchy); //хрупкаво е
67. cout << "Good for eating: "; out(eating); //става за ядене
68. cout << endl;
69. }
70. void Orange::show()
71. {
72. cout << name << " orange is: " << "\n";
73. cout << "Annual: "; out(annual); //едногодишно
74. cout << "Perennial: "; out(perennial); //многогодишно
75. cout << "Tree: "; out(tree); //расте на дърво
76. cout << "Tropical: "; out(tropical); //тропично
77. cout << "Color: " << c[clr] << "\n"; //цвят
78. cout << "Good for juice: "; out(juice); //сочно
79. cout << "Sour: "; out(sour); //кисело
80. cout << "Good for eating: "; out(eating); //става за ядене
81. cout << "\n";
82. }
83. void out(enum yn x)
84. {
85. if(x==no) cout << "no\n";
86. else cout << "yes\n";
87. }
88. int main()
89. {
90. Apple a1, a2;
91. Orange o1, o2;
92. a1.seta("Red Delicious", red, no, yes, yes);
93. a2.seta("Jonathan", red, yes, no, yes);
94. o1.seto("Navel", orange, no, no, yes);
95. o2.seto("Valencia", orange, yes, yes, no);
96. a1.show();
97. a2.show();
98. o1.show();
99. o2.show();
100. return 0;
101. }

Програмата демонстрира нещо важно за наследяването: базовият клас не принадлежи единствено на един производен клас, а може да бъде наследен от произволен брой класове.

Задача 1

Да се напише програма, която дефинира клас People, определящ човек по име и единен граждански номер (ЕГН), а също производен клас Student на класа People, който определя понятието студент като човек, който има факултетен номер и среден успех. Да се дефинира обект от клас Student и се изведе дефинираният обект.

Задача 2

Съставете базов клас триъгълник с четири елемента дължина на съответните страни и конструктори, който инициализират и валидират входните данни. Направете функция на базовия клас за намиране на лицето на триъгълника по хереоновата хормула. Създайте:  
– Производен клас „равнобедрен триъгълник“ и предефинирайте конструктора.  
– Производен клас „равностранен триъгълник“ и предефинирайте конструктора. Напишете втора функция за изчисляване на лицето на триъгълник, която използва директно формулата за лице на равностранен триъгълник: a\*a\*sqrt(3)/4.