3. Псевдоними.

Псевдонимът представлява неявен указател, който играе ролята на друго име за дадена променлива. Съществуват 3 начина, по които може да се използва даден псевдоним:

- Псевдонимът може да бъде предаван на функция
- Псевдонимът може да бъде връщан като резултат от функция
- Може да бъде създаван независим псевдоним

1. Псевдоним като параметър на функция

Когато един обект се предава като параметър на функция посредством подразбиращото се обръщение по стойност, се създава копие на този обект. Въпреки, че конструкторът на параметъра не се извиква, все пак се извиква неговият деструктор, когато функцията върне резултат. В някои случаи това може да доведе до сериозни проблеми – напр., ако деструкторът освобождава някаква заделена памет.

Едно от решенията за този проблем е обекти да се предават чрез **псевдоними**. Когато обекта се подава чрез псевдоним, не се създава копие на обекта, а следователно не се извиква и деструкторът, когато функцията върне резултат. Промените, които се направят с обекта в рамките на функцията се запазват и след излизането от функцията. Ако обектите не трябва да бъдат променени от функцията, то се използват **константни псевдоними.**

Псевдонимът не е указател. Следователно, когато един обект се предава чрез псевдоним, операторът за достъп до член си остава точка(.), а не стрелка(->).

Задача 1. Предаване на константен псевдоним на обект като параметър на функция.

```
#include<iostream.h>
#include<math.h>

class point             //Клас точка
{  private:
     double x;
     double y;
   public:
     point(double xcoord, double ycoord);            //Конструктор с два параметъра
     void print();                                   //Печат на точка
     double dist(const point &p1, const point &p2); //Разстояние между две точки
     ~point();                                      //Деструктор
};
point::point(double xcoord, double ycoord)  //Конструктор с два параметъра
{
  cout << "Creating Point(" << xcoord << "," << ycoord << ")" << endl;
  x = xcoord;
  y = ycoord;
}
void point::print()            //Печат на точка
{
  cout << "Point(" << x << "," << y << ")";
}
double point::dist(const point &p1, const point &p2)      // Разстояние между две точки
{
  return sqrt((p1.x-p2.x)*(p1.x-p2.x)+(p1.y-p2.y)*(p1.y-p2.y));
}
point::~point()  //Деструктор
{ cout << "Destructing ";
  print();
  cout << endl;
}
void main()
{ point a(3,4), b(10,4);      //Обекти от тип Точка
  cout << "Distance between "; a.print();
  cout << " and "; b.print();
  cout << " is " << a.dist(a,b) << endl;
}
```

Резултати:

```
Създава се Точка(3,4)
Създава се Точка(10,4)
Разстоянието между Точка(3,4) и Точка(10,4) е 7
Разрушава се Точка(10,4)
Разрушава се Точка(3,4)
```

2. Псевдоним на обект като върнат резултат от функция.
Функция може да върне псевдоним на обект като резултат. Функция, която връща псевдоним като резултат, може да се използва и от лявата страна на оператора за присвояване.

Задача 2. Следната програма създава клас за масив, който автоматично прави проверка за границите на масива. Класа за масив съдържа 2 основни функции – една, която съхранява информацията от масива, и друга, която извлича информацията. Тези функции могат да проверяват по време на изпълнение дали не се надхвърлят границите на масива.

```
#include<iostream.h>
#include<stdlib.h>

class array {
     int size;
     char *p;
  public:
     array(int num);
     ~array() { delete [] p; }
     char &put(int i);
     char get(int i);
};

array::array(int num)
{
  p = new char [num];
  if(!p){
     cout << ”Allocation error\n”;
     exit(1);
  }
  size = num;
}
char &array::put(int i)   //Поставя се нещо в масива
{
  if(i<0 || i>=size) {
    cout << ”Bounds error!!!\n”;
    exit(1);
  }
  return p[i];           //връща псевдоним за p[i]
}
char array::get(int i)   //Взема се нещо от масива.
{
 if(i<0 || i>=size) {
    cout << ”Bounds error!!!\n”;
    exit(1);
  }
  return p[i];           //връща символ
}

int main()
{
  array a(10);

  a.put(3) = ’X’;
  a.put(2) = ’R’;

  cout << a.get(3) << ”\n” << a.get(2);
  cout << ”\n”;

  a.put(11) = ’!’;      //сега се генерира грешка по време на изпълнение
  return 0;
}
```

Функцията **put()** връща псевдоним към елемента на масива, указан от параметъра **i**. Този псевдоним след това може да бъде използван от лявата страна на конструкция за присвояване, за да съхрани нещо в масива – ако позицията, посочена от **i**, не е извън границите на масива. Противоположната функция е **get()** – тя връща стойността, съхранена на позицията посочена от **i**, ако тя не е извън границите на масива.
В програмата се заделя динамично памет за масива с помощта на **new**. Това дава възможност да се декларират масиви с различни дължини.

Задачи за самостоятелна работа:

Напишете клас Stack, който изобразява стек. Напишете функцията функцията loadstack(), която връща като резултат стек, напълнен със символите от азбуката (a-z). Предефинирайте функцията така, че да приема като параметър стойност от тип int с име upper. Ако upper е 1, попълнете стека с голями букви от азбуката. В противен случай го попълнете с малки букви.