目录

[1 1](#_Toc163215315)

[2 5](#_Toc163215316)

[3 11](#_Toc163215317)

[4 15](#_Toc163215318)

[5 21](#_Toc163215319)

# 1

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  typedef long long ll;  class intArray  {      int csize;      int \*ptr;      int size;  public:      intArray(int size) : size(size)      {          ptr = new int[size];          csize = 0;      }      intArray(const intArray &a)      {          csize = a.csize;          size = a.size;          ptr = new int[size];          for (int i = 0; i < csize; i++)              ptr[i] = a.ptr[i];      }      ~intArray()      {          delete[] ptr;      }      bool input(int x)      {          if (csize == size)              return false;          else          {              ptr[csize++] = x;              return true;          }      }      bool earse(int x)      {          int i;          int mark = 0;          for (i = 0; i < csize; i++)          {              if (ptr[i] == x)              {                  mark = 1;                  break;              }          }          if (mark)          {              for (int j = i; j < csize - 1; j++)              {                  ptr[j] = ptr[j + 1];              }              csize--;          }          if (mark)              return true;          else              return false;      }      void display()      {          for (int i = 0; i < csize; i++)              cout << ptr[i] << ' ';      }      int find(int x)      {          int mark = 0;          for (int i = 0; i < csize; i++)          {              if (x == ptr[i])                  mark++;          }          return mark;      }      void becomeBigger()      {          int \*temp = new int[2 \* size];          for (int i = 0; i < csize; i++)              temp[i] = ptr[i];          delete[] ptr;          ptr = new int[2 \* size];          for (int i = 0; i < csize; i++)              ptr[i] = temp[i];          delete[] temp;      }      int Size()      {          return size;      }      int Csize()      {          return csize;      }      int look(int x)      {          if (x >= 0 && x < csize)              return ptr[x];          else              return 0;      }      friend intArray operator+(const intArray &a, const intArray &b);  };  intArray operator+(const intArray &a, const intArray &b)  {      intArray temp(a.size + b.size);      for (int i = 0; i < a.csize; i++)      {          temp.ptr[temp.csize++] = a.ptr[i];      }      for (int i = 0; i < b.csize; i++)      {          temp.ptr[temp.csize++] = b.ptr[i];      }      return temp;  }  int main()  {      ios::sync\_with\_stdio(false);      cin.tie(0);      cout.tie(0);      intArray a(5);      intArray b(3);      for (int i = 5; i <= 8; i++)          a.input(i);      for (int i = 100; i <= 102; i++)          b.input(i);      cout << "a is ";      a.display();      cout << endl;      cout << "b is ";      b.display();      cout << endl;      intArray c = a + b;      cout << "c is ";      c.display();      cout << endl;      c.earse(5);      c.earse(100);      cout << "c is ";      c.display();      cout << endl;      if (c.find(6))          cout << "find it 6" << endl;      else          cout << "404 not find 6" << endl;      if (c.find(10086))          cout << "find it 10086" << endl;      else          cout << "404 not find 10086" << endl;      return 0;  } |
|  |
|  |

# 2

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  typedef long long ll;  class intArray  {      int csize;      int \*ptr;      int size;  public:      intArray(int size) : size(size)      {          ptr = new int[size];          csize = 0;      }      intArray(const intArray &a)      {          csize = a.csize;          size = a.size;          ptr = new int[size];          for (int i = 0; i < csize; i++)              ptr[i] = a.ptr[i];      }      ~intArray()      {          delete[] ptr;      }      bool input(int x)      {          if (csize >= size)              return false;          else          {              ptr[csize++] = x;              return true;          }      }      bool earse(int x)      {          int i;          int mark = 0;          for (i = 0; i < csize; i++)          {              if (ptr[i] == x)              {                  mark = 1;                  break;              }          }          if (mark)          {              for (int j = i; j < csize - 1; j++)              {                  ptr[j] = ptr[j + 1];              }              csize--;          }          if (mark)              return true;          else              return false;      }      void display()      {          for (int i = 0; i < csize; i++)              cout << ptr[i] << ' ';      }      int find(int x)      {          int mark = 0;          for (int i = 0; i < csize; i++)          {              if (x == ptr[i])                  mark++;          }          return mark;      }      void becomeBigger()      {          int \*temp = new int[2 \* size];          for (int i = 0; i < csize; i++)              temp[i] = ptr[i];          delete[] ptr;          ptr = new int[2 \* size];          for (int i = 0; i < csize; i++)              ptr[i] = temp[i];          delete[] temp;          size \*= 2;      }      int Size()      {          return size;      }      int Csize()      {          return csize;      }      int look(int x)      {          if (x >= 0 && x < csize)              return ptr[x];          else              return 0;      }      friend intArray operator+(const intArray &a, const intArray &b);  };  intArray operator+(const intArray &a, const intArray &b)  {      intArray temp(a.size + b.size);      for (int i = 0; i < a.csize; i++)      {          temp.ptr[temp.csize++] = a.ptr[i];      }      for (int i = 0; i < b.csize; i++)      {          temp.ptr[temp.csize++] = b.ptr[i];      }      return temp;  }  class intSet : public intArray  {  public:      intSet(int s) : intArray(s)      {      }      intSet(const intSet &a) : intArray(a)      {          unique();      }      void unique()      {          for (int i = 0; i < intArray::Csize(); i++)          {              int wei = intArray::find(intArray::look(i));              wei--;              while (wei--)                  intArray::earse(intArray::look(i));          }      }      void insert(int x)      {          if (!input(x))          {              intArray::becomeBigger();              input(x);          }          unique();      }      void print()      {          for (int i = 0; i < intArray::Csize(); i++)          {              cout << intArray::look(i) << ' ';          }      }      friend intSet operator+(intSet &a, intSet &b);      friend intSet operator\*(intSet &a, intSet &b);  };  intSet operator+(intSet &a, intSet &b)  {      intSet temp = a;      for (int i = 0; i < b.Csize(); i++)      {          if (!temp.input(b.look(i)))          {              temp.becomeBigger();              temp.input(b.look(i));          }      }      temp.unique();      return temp;  }  intSet operator\*(intSet &a, intSet &b)  {      intSet temp = a;      for (int i = 0; i < a.Csize(); i++)      {          if (!b.find(a.look(i)))          {              temp.earse(a.look(i));          }      }      temp.unique();      return temp;  }  int main()  {      ios::sync\_with\_stdio(false);      cin.tie(0);      cout.tie(0);      intSet a(3);      a.insert(3);      a.insert(3);      a.insert(3);      a.insert(4);      a.insert(5);      a.insert(6);      cout << "a is :" << endl;      a.print();      cout << endl;      intSet b(3);      b.insert(5);      b.insert(5);      b.insert(6);      b.insert(6);      b.insert(100);      b.insert(102);      cout << "b is " << endl;      b.print();      cout << endl;      intSet c = a + b;      cout << "bingji" << endl;      c.print();      cout << endl;      intSet d = a \* b;      cout << "jiaoji" << endl;      d.print();      return 0;  } |
|  |
|  |

# 3

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  typedef long long ll;  struct Point  {      long double x;      long double y;  };  class Shape  {  private:      Point center;  public:      Shape(long double x, long double y)      {          center.x = x;          center.y = y;      }      Shape(Point x)      {          center.x = x.x;          center.y = x.y;      }      Shape(const Shape &a)      {          center.x = a.center.x;          center.y = a.center.y;      }      Shape()      {          center.x = 0;          center.y = 0;      }      long double getcenterx()      {          return center.x;      }      long double getcentery()      {          return center.y;      }  };  class Squre : public Shape  {      long double bianchang;  public:      Squre(long double x, long double y, long double bc)          : Shape(x, y), bianchang(bc) {}      Squre() : Shape(), bianchang(0) {}      Squre(const Squre &a)          : Shape(a), bianchang(a.bianchang) {}      long double c()      {          return bianchang \* 4;      }      long double s()      {          return bianchang \* bianchang;      }      int isin(long double xx, long double yy)      {          if (getcenterx() + bianchang / 2.0 > xx && getcenterx() - bianchang / 2.0 < xx && getcentery() + bianchang / 2.0 > yy && getcentery() - bianchang / 2.0 < yy)              return -1;          else if (getcenterx() + bianchang / 2.0 < xx || getcenterx() - bianchang / 2.0 > xx || getcentery() + bianchang / 2.0 < yy || getcentery() - bianchang / 2.0 > yy)              return 1;          else              return 0;      }  };  class Circle : public Shape  {      long double bianchang;  public:      Circle(long double x, long double y, long double bc)          : Shape(x, y), bianchang(bc) {}      Circle() : Shape(), bianchang(0) {}      Circle(const Circle &a)          : Shape(a), bianchang(a.bianchang) {}      long double c()      {          return bianchang \* 2 \* 3.1415926535;      }      long double s()      {          return bianchang \* bianchang \* 3.1415926535;      }      int isin(long double xx, long double yy)      {          if ((xx - getcenterx()) \* (xx - getcenterx()) + (yy - getcentery()) \* (yy - getcentery()) < bianchang \* bianchang)              return -1;          else if ((xx - getcenterx()) \* (xx - getcenterx()) + (yy - getcentery()) \* (yy - getcentery()) == bianchang \* bianchang)              return 0;          else              return 1;      }  };  int main()  {      ios::sync\_with\_stdio(false);      cin.tie(0);      cout.tie(0);      Squre fang(1, 1, 2);      cout << fang.c() << endl;      cout << fang.s() << endl;      if (fang.isin(0, 0) == -1)          cout << "在里面" << endl;      else if (fang.isin(0, 0) == 0)          cout << "边上" << endl;      else if (fang.isin(0, 0) == 1)          cout << "外面" << endl;      if (fang.isin(1, 1) == -1)          cout << "在里面" << endl;      else if (fang.isin(1, 1) == 0)          cout << "边上" << endl;      else if (fang.isin(1, 1) == 1)          cout << "外面" << endl;      if (fang.isin(10, 0) == -1)          cout << "在里面" << endl;      else if (fang.isin(10, 0) == 0)          cout << "边上" << endl;      else if (fang.isin(10, 0) == 1)          cout << "外面" << endl;      Circle yuan(1, 1, 1);      cout << yuan.c() << endl;      cout << yuan.s() << endl;      if (yuan.isin(0, 1) == -1)          cout << "在里面" << endl;      else if (yuan.isin(0, 1) == 0)          cout << "边上" << endl;      else if (yuan.isin(0, 1) == 1)          cout << "外面" << endl;      if (yuan.isin(1, 1) == -1)          cout << "在里面" << endl;      else if (yuan.isin(1, 1) == 0)          cout << "边上" << endl;      else if (yuan.isin(1, 1) == 1)          cout << "外面" << endl;      if (yuan.isin(10, 1) == -1)          cout << "在里面" << endl;      else if (yuan.isin(10, 1) == 0)          cout << "边上" << endl;      else if (yuan.isin(10, 1) == 1)          cout << "外面" << endl;      return 0;  } |
|  |
|  |

# 4

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  typedef long long ll;  class myString  {  public:      char \*str;      int size;      myString(const char \*s)      {          size = strlen(s);          str = new char[size + 1];          strcpy(str, s);          str[size] = '\0';      }      myString()      {          size = 0;          str = NULL;      }      ~myString()      {          delete[] str;      }      friend ostream &operator<<(ostream &out, const myString &a);      friend istream &operator>>(istream &in, myString &a);      myString &      operator=(const myString &a)      {          if (str != NULL)              delete[] str;          size = a.size;          str = new char[size + 1];          strcpy(str, a.str);          str[size] = '\0';          return \*this;      }      myString operator+(const myString &a)      {          char \*temp = new char[a.size + this->size + 1];          strcpy(temp, this->str);          strcat(temp, a.str);          temp[a.size + this->size] = '\0';          myString re(temp);          return re;      }      int cmp(const myString &a)      {          return strcmp(this->str, a.str);      }  };  ostream &operator<<(ostream &out, const myString &a)  {      for (int i = 0; i < a.size; i++)          out << a.str[i];      return out;  }  istream &operator>>(istream &in, myString &a)  {      if (a.str != NULL)          delete[] a.str;      char temp[100006];      in >> temp;      int wei = 0;      while (temp[wei] != '\0')          ++wei;      a.str = new char[wei + 1];      a.size = wei + 1;      int i = 0;      for (i = 0; i < wei; i++)          a.str[i] = temp[i];      a.str[i] = '\0';      return in;  }  class EditString : public myString  {      int position;  public:      EditString(const char \*s) : myString(s)      {          position = 0;      }      EditString(const myString &s) : myString(s.str)      {          position = 0;      }      EditString() : myString()      {          position = 0;      }      EditString(const EditString &a) : myString(a.str), position(a.position) {}      EditString son(int a)      {          char \*temp = new char[size - a + 1];          int wei = 0;          for (int i = a; i < this->size; i++)              temp[wei++] = this->str[i];          temp[wei] = '\0';          myString re(temp);          delete[] temp;          return re;      }      EditString insert(int a, const char \*s)      {          int i;          int newlength = size + strlen(s);          char \*temp = new char[strlen(s) + size + 1];          for (i = 0; i < a; i++)          {              temp[i] = str[i];          }          int j = 0;          while (s[j] != '\0')          {              temp[i++] = s[j++];          }          for (int k = a; k < size; k++)          {              temp[i++] = str[k];          }          temp[newlength] = '\0';          delete[] str;          str = new char[newlength + 1];          strcpy(str, temp);          str[newlength] = '\0';          size = newlength;          delete[] temp;          return \*this;      }      EditString change(int a, const char \*s)      {          int i;          int newlength = a + strlen(s);          char \*temp = new char[newlength + 1];          for (i = 0; i < a; i++)              temp[i] = str[i];          int j = 0;          while (s[j] != '\0')              temp[i++] = s[j++];          temp[newlength] = '\0';          delete[] str;          size = newlength;          str = new char[newlength + 1];          strcpy(str, temp);          str[newlength] = '\0';          delete[] temp;          return \*this;      }      EditString backspace(int a)      {          char \*temp = new char[a + 1];          for (int i = 0; i < a; i++)          {              temp[i] = str[i];          }          delete[] str;          str = new char[a + 1];          strcpy(str, temp);          str[a] = '\0';          size = a;          delete[] temp;          return \*this;      }      friend ostream &operator<<(ostream &out, const EditString &a);  };  ostream &operator<<(ostream &out, const EditString &a)  {      for (int i = 0; i < a.size; i++)          out << a.str[i];      return out;  }  int main()  {      ios::sync\_with\_stdio(false);      cin.tie(0);      cout.tie(0);      myString s1, s2;      cout << "输入s1" << endl;      cin >> s1;      cout << "s1 is " << s1 << endl;      cout << "输入s2" << endl;      cin >> s2;      cout << "s2 is " << s2 << endl;      cout << "把s2赋给s1" << endl;      s1 = s2;      cout << "s1 is " << s1 << endl;      cout << "输入s1" << endl;      cin >> s1;      cout << "输入s2" << endl;      cin >> s2;      cout << "s3是s1+s2" << endl;      myString s3 = s1 + s2;      cout << "s3是 " << s3 << endl;      cout << "比较s1和s2" << endl;      if (s1.cmp(s2) < 0)          cout << "s1 small" << endl;      else if (s1.cmp(s2) == 0)          cout << "s1 = s2" << endl;      else          cout << "s1 is big" << endl;      cout << "一个叫abcdefg的字符串" << endl;      EditString a("abcdefg");      cout << a.str << endl;      cout << "子串2" << endl;      cout << a.son(2) << endl;      cout << "第二位插入lcx" << endl;      cout << a.insert(2, "lcx") << endl;      cout << "第三位之后变成QAQ" << endl;      cout << a.change(3, "QAQ") << endl;      cout << "把第四位之后的删除" << endl;      cout << a.backspace(4) << endl;      return 0;  } |
|  |
|  |

# 5

|  |
| --- |
| #include <bits/stdc++.h>  using namespace std;  typedef long long ll;  class yuanGong  {  protected:      int id;      string name;      double money;  public:      yuanGong(int idd, string namee) : id(idd), name(namee), money(0) {}      virtual void calculate() = 0;      void show()      {          cout << "id " << id << " name " << name << " money " << money << endl;      }  };  class jingLi : virtual public yuanGong  {  public:      jingLi(int idd, string namee) : yuanGong(idd, namee) {}      void calculate()      {          money = 8000;      }  };  class jiShuRenRuan : virtual public yuanGong  {  protected:      int hour;  public:      jiShuRenRuan(int id, string name, int hour) : yuanGong(id, name), hour(hour) {}      void calculate()      {          money = hour \* 100;      }  };  class xiaoShouYuan : virtual public yuanGong  {  protected:      double xiaoshouLiang;  public:      xiaoShouYuan(int id, string name, double xiao) : yuanGong(id, name), xiaoshouLiang(xiao) {}      void calculate()      {          money = xiaoshouLiang / 100.0 \* 4.0;      }  };  class xiaoshoujingli : public xiaoShouYuan, public jingLi  {  public:      xiaoshoujingli(int id, string name, double xiao) : yuanGong(id, name), xiaoShouYuan(id, name, xiao), jingLi(id, name) {}      void calculate()      {          money = xiaoshouLiang / 1000.0 \* 5.0 + 5000;      }  };  int main()  {      ios::sync\_with\_stdio(false);      cin.tie(0);      cout.tie(0);      cout << "月工资计算办法是：经理拿固定月薪 8000 元，兼职技术人员按每小时 100 元领取月薪，销售员按当月销售额的 4 %提成，销售经理既拿固定月工资也领取销售提成，固定月工资为 5000 元，销售提成为所管辖部门当月销售总额的 5‰ "<<endl;      cout << "技术人员干了1000h，销售卖了100000,销售经理卖了1000" << endl;      jingLi jl(123, "woshijingli");      jiShuRenRuan jsry(666, "jishurenyuan", 1000);      xiaoShouYuan xsry(114514, "xiaoshou", 100000);      xiaoshoujingli xsjl(233, "xiaoshoujingliAAA", 1000);      jl.calculate();      jsry.calculate();      xsry.calculate();      xsjl.calculate();      jl.show();      jsry.show();      xsry.show();      xsjl.show();      return 0;  } |
|  |
|  |