| **C++** | **JAVA** |
| --- | --- |
| string  ● size()  ● [ index ]  ● compare(elem) ó “==”  ● substr(beginIndex)  ● substr(beginIndex,**substringLength**)  ● find(elem)  ● operador + para concatenación  ● Adicionales:  ○ swap  ○ c\_str  ○ replace  ○ rfind  ○ find\_first\_of  ○ find\_last\_of  ○ find\_first\_not\_of  ○ find\_last\_not\_of  ○ erase  ○ insert  Nota: en c++ se pueden modificar fácilmente las cadenas, en Java se tendría que hacer vía StringBuilder y/o llamadas a substring mas concatenaciones;cosas como las que hace insert y erase | String  ● length()  ● charAt(index)  ● compareTo(elem) NUNCA USEN “=” (= compara referencias a objetos)  ● substring(beginIndex)  ● substring(beginIndex,**endIndex**)  ● indexOf(elem)  ● operador + para concatenación  ● Adicionales:  ○ compareToIgnoreCase(str)  ○ contains(str)  ○ contentEquals(str)  ○ endsWith(str)  ○ equalsIgnoreCase(str)  ○ replace(...)  ○ replaceFirst(str,nuevaStr)  ○ **split(String regex)**  ○ startsWith(String prefix)  ○ substring(int beginIndex)  ○ toUpperCase()  ○ **trim()**  ○ lastIndexOf(...)  Nota: split() y trim() son funciones muy útiles en competencias de programación que no tiene C++. |
| vector  ● push\_back(elem)  ● size()  ● [ index ]  ● insert(index,elem)  ● [ index ] = elem  ● clear()  ● erease(index) | ArrayList  ● add(elem)  ● size()  ● get(index)  ● add(index,elem)  ● set(index,elem)  ● clear()  ● remove(index) |
| deque  Accesos:  operator[]  at | LinkedList  Accesos:  E element()  E get(int index) |

| front  back  Modificadores  push\_back  push\_front  pop\_back  pop\_front  insert  erase  clear  swap | E getFirst()  E getLast()  E getFirst()  E getLast()  E peek()  E peekFirst()  E peekLast()  int size()  Modificadores:  boolean offer(E e)  boolean offerFirst(E e)  boolean offerLast(E e)  E poll()  E pollFirst()  E pollLast()  E pop()  void push(E e)  E remove()  E removeFirst()  void addFirst(E e)  void addLast(E e)  void clear() |
| --- | --- |
| list  Capacidad:  empty  size  Accesos:  front  back  Modificadores:  assign  push\_front  pop\_front  push\_back  pop\_back  insert | LinkedList |

| erase  swap  clear  Operaciones:  splice  remove  remove\_if  unique  merge  sort  reverse |  |
| --- | --- |
| queue  empty  size  front  back  push  pop | LinkedList |
| set  Consultas de tamaño:  empty  size  Modificaciones:  insert  erase  swap  clear  Operaciones:  find  count | TreeSet o HashSet  TreeSet  boolean add(E e)  boolean addAll(Collection<? extends E> c) E ceiling(E e)  void clear()  Object clone()  boolean contains(Object o)  E first()  E floor(E e)  boolean isEmpty()  E last()  E lower(E e)  E pollFirst()  E pollLast()  boolean remove(Object o)  int size() |

|  | HashSet:  boolean add(E e)  void clear()  Object clone()  boolean contains(Object o)  boolean isEmpty()  boolean remove(Object o)  int size() |
| --- | --- |

| map  Capacidad:  empty  size  Element access:  operador[] para acceso  Modificadores  insert  erase  swap  clear  Operaciones:  find  count | TreeMap o HashMap  TreeMap  boolean containsKey(Object key)  boolean containsValue(Object value)  V get(Object key)  NavigableSet<K> descendingKeySet()  NavigableMap<K,V> descendingMap()  Set<Map.Entry<K,V>> entrySet()  Set<K> keySet()  Map.Entry<K,V> lastEntry()  K lastKey()  V put(K key, V value)  void putAll(Map<? extends K,? extends V> map) V remove(Object key)  int size()  Collection<V> values()  HashMap  void clear()  Object clone()  boolean containsKey(Object key)  boolean containsValue(Object value)  Set<Map.Entry<K,V>> entrySet()  V get(Object key)  boolean isEmpty()  Set<K> keySet()  V put(K key, V value)  void putAll(Map<? extends K,? extends V> m) V remove(Object key)  int size()  Collection<V> values() |
| --- | --- |
| stack  empty  size  top  push  pop | Stack  boolean empty()  E peek()  E pop()  E push(E item)  int search(Object o) |

| bitset  operador [] para acceso  Operaciones de bits:  set  reset  flip  Operaciones del bitset:  to\_ulong  to\_string  count  size  test  any  none | BitSet  void and(BitSet set)  void andNot(BitSet set)  int cardinality()  void clear()  void clear(int bitIndex)  void clear(int fromIndex, int toIndex)  Object clone()  boolean equals(Object obj)  void flip(int bitIndex)  void flip(int fromIndex, int toIndex)  boolean get(int bitIndex)  BitSet get(int fromIndex, int toIndex) boolean intersects(BitSet set)  boolean isEmpty()  int length()  int nextClearBit(int fromIndex) index.  int nextSetBit(int fromIndex)  void or(BitSet set)  void set(int bitIndex)  void set(int bitIndex, boolean value)  void set(int fromIndex, int toIndex)  void set(int fromIndex, int toIndex, boolean value) int size()  String toString()  void xor(BitSet set) |
| --- | --- |
| multimap  Capacidad:  empty  size  Modificadores:  insert  erase  swap  clear  Operaciones: | ----- |

| find  count |  |
| --- | --- |
| multiset  Capacidad:  empty  size  Modificdores:  insert  erase  swap  clear  Operaciones:  find  count | ----- |
| pair<anyType,anyType>  Ejemplo, parejas de enteros:  pair<int,int> elem = make\_pair(1,2);  se accede con elem.first y elem.second, al ordenarlos se ordena primero por .first y luego por .second | static class Pair implements Comparable<Pair>{ int first,second;  public Pair(int first, int second){  this.first = first;  this.second = second;  }  public String toString(){  return "("+first+","+second+")";  }  public boolean equals( Object o ){  if( !(o instanceof Pair))return false;  return this.compareTo((Pair)o) == 0;  }  /\*retorna -1 si this va antes que “o” 1 si “o” va después y 0 si son iguales\*/  public int compareTo(Pair o){  if( this.first < o.first )return -1;  if( this.first > o.first )return 1;  if( this.second < o.second )return -1;  if( this.second > o.second )return 1;  return 0;  } |

|  | public Pair clone(){  return new Pair(first , second);  }  public int hashCode(){  return (first+""+second).hashCode();  }  }  Nota:Cuando se van a almacenar en TreeSet o TreeMap, debe tener la interfaz Comparable y el método compareTo e equals, y cuando se va a almacenar en un HashSet o HashMap, deben tener los métodos hashcode y equals. Y porsupuesto que viola todo POO. |
| --- | --- |
| Ordenamiento en C++:  sort(elem.begin(),elem.end()) | Ordenamiento en java  Arrays.sort(algunArray) ó Collections.sort(list) |
| sort(elem.begin(),elem.end(),cmp\_function)  Crear antes la función:  bool cmp\_function( const anyType& a , const anyType& b)  {/\*retorna true cuando a < b\*/}  para ordenar al revés:  sort(elem.rbegin(),elem.rend()); | Crear clase que implemente  Comparable:  class MyClass implements Comparable<MyClass>{...public int compareTo(MyClass o){...}...}  y después usar Arrays.sort(algunArray) o  Collections.sort(AlgunaCollection);  o crear un comparator:  class MyComparator implements  Comparator<MyClass>{...public int compare(MyClass a , MyClass b){...}}  y después usar  Arrays.sort(algunArray,instanciaDelComparador) o Collections.sort(list,instanciaDelComparador);  Para ordenar al revés, se debe agregar el parámetro |
| priority\_queue<E>  ● size  ● empty | PriorityQueue<E>  ● boolean add(E o)  ● void clear() |

| ● pop  ● push  ● top | ● boolean offer(E o)  ● E peek()  ● E poll()  ● boolean remove(Object o)  ● int size() |
| --- | --- |
|  |  |
| next\_permutation(elem.begin(),elem.end()) | boolean next\_permutation(int[] vec)  {  int tmp;  for(int i=vec.length-2; i>=0; i--)  if(vec[i+1]>vec[i])  for(int j=vec.length-1; j>=0; j--)  if(vec[j]>vec[i])  {  tmp=vec[i];  vec[i]=vec[j];  vec[j]=tmp;  for(int k=i+1,l=vec.length-1;k<l;k++,l--) {  tmp=vec[k];  vec[k]=vec[l];  vec[l]=tmp;  }  return true;  }  return false;  } |
| prev\_permutation(elem.begin(),elem.end()) | ----- |
| Llenado de arreglos en C++  memset(0,arreglo,sizeof(arreglo));  memset(-1,arreglo,sizeof(arreglo));  memset(false,arreglo,sizeof(arreglo));  fill(arreglo,arreglo+tamArreglo,valor);  fill(iteradorInicio,iteradorFinal,valor); | Llenado de arreglos en Java  Arrays.fill(0,arreglo);  Arrays.fill(-1,arreglo);  Arrays.fill(false,arreglo);  Arrays.fill(cualquierValor,arreglo);  Collections.fill(algunaLista,valor); |

| Nota:memset sirve para arreglos de una o varias dimensiones y es muy rápido, pero se recomienda usar solo para arreglos booleanos a falso y para enteros a 0 o -1 debido a que es una función diseñada para arreglos de caracteres. fill sirve tanto para contenedores como para arreglos unidimensionales. | Nota: fill solo funciona para arreglos de una dimensión. Para llenar arreglos de mas de una dimensión se debe hacer con bucles anidados. |
| --- | --- |
| \_\_gcd(a,b) | BigInteger bigA = new BigInteger(a+””);  y luego:  bigA.gcd(new BigInteger(b+””)).intValue(); ó  bigA.gcd(new BigInteger(b+””)).longValue(); |
| Algorithm de C++  min\_element  max\_element  min\_element  max\_element  binary\_search  reverse  rotate  random\_shuffle  iter\_swap  copy  count  Otros solo de C++:  count\_if  find  find\_first\_of  find\_if  for\_each  lexicographical\_compare  merge  partition  prev\_permutation  replace | Collections de Java  Collections.min(list)  Collections.max(list)  Collections.min(list,comparador)  Collections.max(list,comparador)  Collections.binarySearch(listaOrdenada)  Collections.reverse(list)  Collections.rotate(list, distance)  Collections.shuffle(list)  Collections.swap(list,i,j)  Collections.copy(listaDestino,listaOrigen) Collections.frequency(list,objeto)  Otros solo de Java:  Collections.disjoint(list1,list2)  System.arraycopy(Object src, int srcPos, Object dest, int destPos, int length)  Mas Utilidades en Collections |

| replace\_if  search  set\_difference  set\_intersection  set\_symmetric\_difference  set\_union  swap  unique  partition  Mas Utilidades y detalles de las funciones anteriores en la librería algorithm |  |
| --- | --- |
| complex  complex::imag  complex::real  Operadores sobrecargados:  =  +=  -=  \*=  /=  +  -  \*  /  ==  != | ----- |
| ----- | BigInteger  Para enteros muy grandes, operaciones básicas soportadas, mas:  ● modInverse(BigInteger m)  ● modPow(exponent,m) |
| ----- | BigDecimal: Para operaciones con números reales, cuando la precisión de double no basta. |
| ----- | StringBuilder: Para Manipular cadenas de caracteres eficientemente, muy útil sobre todo cuando se requiere “imprimir” muchas cosas. |

| ----- | StringTokenizer: Para tokenizar (separar) cadenas. |
| --- | --- |
| ctype  isalnum  isalpha  iscntrl  isdigit  isgraph  islower  isprint  ispunct  isspace  isupper  isxdigit  tolower  toupper | Character:  int compareTo(Character anotherCharacter) int digit(char ch, int radix)  int digit(int codePoint, int radix)  boolean equals(Object obj)  char forDigit(int digit, int radix)  int getNumericValue(char ch)  int getNumericValue(int codePoint)  boolean isDefined(char ch)  boolean isDigit(char ch)  boolean isJavaLetter(char ch)  boolean isLetter(char ch)  boolean isLetterOrDigit(char ch)  boolean isLowerCase(char ch)  boolean isSpace(char ch)  boolean isSpaceChar(char ch)  boolean isUpperCase(char ch)  boolean isWhitespace(char ch)  char toLowerCase(char ch)  char toUpperCase(char ch)  Character valueOf(char c) |
| Función trim para eliminar “espacios” al inicio y al final de las cadenas en C++  void trim( string& c )  {  if( c.size() == 0 )  return;int i , p = -1 , q = -1;  for( i = 0; i < c.size(); ++ i )  if( int(c[i])%255 > int(' ') )  {  p = i;  break;  }  for( i = c.size()-1; i >= 0; -- i )  if( int(c[i])%255 > int(' ') )  {  q = i; | Función trim para eliminar “espacios” al inicio y al final de las cadenas en Java  algunaCadena.trim() |

| break;  }  c = p == -1 ? "" : c.substr(p,q-p+1);  }  inline string trimm( string c )  {  string s(c);  trim(s);  return s;  } |  |
| --- | --- |
| Tokenización en C++  template<class T> vector<T> strtovt(string s){ vector<T> ret;istringstream f(s);  T tmp;  while (f >> tmp)  ret.push\_back(tmp);  return ret;  }  vector<string> tokenize(const string& str, const string& d = " "){  vector <string> t;  int up = str.find\_first\_not\_of(d, 0);  int pos = str.find\_first\_of(d, up);  while (string::npos != pos || string::npos != up){ t.push\_back(str.substr(up, pos - up));  up = str.find\_first\_not\_of(d, pos);  pos = str.find\_first\_of(d, up);  }return t;  }  vector<int> splittoint( const string& s,  const string& delim =" " )  {  vector <string> tok = tokenize(s,delim); vector <int> res;  for(int i=0;i<tok.size();++i)  res.push\_back( atoi( tok[i].c\_str() ) );  return res;  } | Tokenización en java  StringTokenizer st = new  StringTokenizer(algunaCadena);  while( st.hasMoreTokens() )  {  String str = st.nextToken();  }  String arr[] = algunaCadena.split(“[ \t\n]+”); |
| /\*para int\*/  #define ones(n) \_\_builtin\_popcount(n) | int onBits = Integer.bitCount(algunInt);  int onBits = Long.bitCount(algunLong); |

| /\*para long long\*/  #define onesL(n) \_\_builtin\_popcountll(n) | int onBits = algunBigInteger.bitCount(); |
| --- | --- |
| printf(“%.3lf”,algunDouble); | System.out.printf(“%.3f”,algunDouble);  ó  static final DecimalFormat FORMATER = new DecimalFormat("0.000");  static String format( double num )  {  return FORMATER.format(num).replaceAll(",", "."); } |
| ----- | Polygon  void addPoint(int x, int y)  boolean contains(x, y)  boolean contains(x, y, w, h)  boolean contains(int x, int y)  boolean contains(Point p)  boolean contains(Point2D p)  Rectangle getBounds()  Rectangle2D getBounds2D()  boolean intersects(x, y, w, h)  void translate(int deltaX, int deltaY) |
| Converción de string a tipo numérico en C++  unsigned long long toi64(string s){  unsigned long long v;  istringstream sin(s);  sin>>v;  return v;  }  template<class T> int toint( T s ){  int v;  istringstream sin( tostring(s) );  sin>>v; | Converción de string a tipo numérico en Java  int n = Integer.parseInt(str);  long n = Long.parseLong(str);  double n = Double.parseDouble(str); |

| return v;  }  inline int s2i( string a ){  return atoi( a.c\_str() );  }  double todouble(string s){  double v;  istringstream sin(s);  sin>>v;  return v;  } |  |
| --- | --- |
| Conversión de tipo numérico a string en C++  template<class T> string tostring(T x){  ostringstream sout;  sout<<x;  return sout.str();  } | Conversión de tipo numérico a String en Java  String str = String.valueOf(cualquierCosa); ó  String str = “”+algunNumero; |
| inline unsigned long int todecimal( string n , int b ) {  return strtoul(n.c\_str(),NULL,b);  }  const string DIGITS  = "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ";  long long todec(string n, int b){  long long r = 0,m = 1;  for (int i = n.length() - 1; i >= 0; --i) r += (long long)DIGITS.find(n[i]) \* m,  m \*= (long long)b;return r;  }  string tobase(int n, int b){  if( n == 0 )return "0";  string r = "";  while (n != 0)  r = tostring (DIGITS[n % b]) + r,  n /= b;return r;  } | static int parseInt(String s, int radix)  static long parseLong(String s, int radix)  static int parseInt(String s, int radix)  static long parseLong(String s, int radix)  static int parseInt(String s, int radix)  static long parseLong(String s, int radix) BigInteger(String val, int radix)//Constructor  static String toBinaryString(int i)  static String toHexString(int i)  static String toOctalString(int i)  static String toString(int i)  static String toString(int i, int radix)  static String toBinaryString(long i)  static String toHexString(long i)  static String toOctalString(long i)  static String toString(long i)  static String toString(long i, int radix) |

| /\*No es estándar:  inline string tobase2( int n , int b )  {  char ccc[100];  return string(itoa (n,ccc,b));  }  \*/ | String toString(int radix) //BigInteger |
| --- | --- |
| M\_E  M\_PI  acos  asin  atan  atan2  ceil  cos  cosh  exp  fabs  floor  fmod  frexp  ldexp  log  log10  modf  pow  sin  sinh  sqrt  tan  tanh | E  PI  abs(num)  acos(num)  asin(num)  atan(num)  atan2(double y, double x)  cbrt(double a)  ceil(double a)  cos(double a)  cosh(double x)  exp(double a)  expm1(double x) Returns ex -1.  floor(double a)  getExponent(double d)  hypot(double x, double y)  IEEEremainder(double f1, double f2)  log(double a)  log10(double a)  log1p(double x)  max(a,b)  min(a,b)  pow(double a, double b)  random()  rint(double a)  round(double a)  sin(double a)  sinh(double x)  sqrt(double a)  tan(double a)  tanh(double x)  toDegrees(double angrad)  toRadians(double angdeg) |
| const int dx[] = {-1,0,0,1}; | static final int dx[] = {-1,0,0,1}; |

| const int dy[] = {0,-1,1,0};  const int dxd[] = {-1,-1,1,1};  const int dyd[] = {-1,1,-1,1};  const int dx8[] = {-1,-1,-1 , 0,0 , 1,1,1}; const int dy8[] = {-1, 0, 1 , -1,1 , -1,0,1}; const int dxc[] = {-2,-2 , -1,-1 , 1,1 , 2,2}; const int dyc[] = {-1, 1 , -2, 2 , -2,2 , -1,1}; cosnt int meses[]  ={0,31,28,31,30,31,30,31,31,30,31,30,31}; | static final int dy[] = {0,-1,1,0};  static final int dxd[] = {-1,-1,1,1};  static final int dyd[] = {-1,1,-1,1};  static final int dx8[] = {-1,-1,-1 , 0,0 , 1,1,1};  static final int dy8[] = {-1, 0, 1 , -1,1 , - 1,0,1};  static final int dxc[] = {-2,-2 , -1,-1 , 1,1 , 2,2};  static final int dyc[] = {-1, 1 , -2, 2 , -2,2 , - 1,1};  static final int meses[]  ={0,31,28,31,30,31,30,31,31,30,31,30,31}; |
| --- | --- |
| long long \_begin = clock();  /\*mi código\*/  cout << (clock() - \_begin)/(CLOCKS\_PER\_SEC/1000)<< " milisegundos "<<endl; | long \_begin = System.currentTimeMillis();    /\*mi código\*/  System.out.println(  (System.currentTimeMillis()-\_begin)+" milisegundos"); |
| Comparador de Doubles  Retorna 0 si se consideran iguales, -1 si x es menor que y o 1 si y es mayor que x. El valor tol debe ser un valor muy pequeño, como 10 e-10.  #define EPS (1e-10)  inline int cmp(const double& x, const double& y = 0, double tol = EPS)  {  return (x <= y + tol) ? (x + tol < y) ? -1 : 0 : 1; } | Comparador de Doubles  Retorna 0 si se consideran iguales, -1 si x es menor que y o 1 si y es mayor que x. El valor tol debe ser un valor muy pequeño, como 10e-10  static int cmp(final double x, final double y , final double eps )  {  return (x <= y + eps) ? (x + eps < y) ? -1 : 0 : 1; } |
| macrodefiniciones  TAD Point  #include<iostream>  #include<string>  #include<cmath>  #include<cstdio>  using namespace std;  /\*Función para comparar dos números reales.( Se puede definir la precisión especificando el épsilon SIEMPRE se debe usar para comparar reales, nunca | -----  **TAD Point**  Point2D -> Point2D.Double ó Point (para coordenadas enteras) Object clone()  distance( px, py)  static distance( x1, y1, x2, y2)  distance(Point2D pt)  distanceSq( px, py)  static distanceSq( x1, y1, x2, y2) |

| usar los operadores == o != ó <= , < , >= , > \*/  const double EPS = 1e-10;  inline int cmp(double x, double y = 0, double tol = EPS){  return (x <= y + tol) ? (x + tol < y) ? -1 : 0 : 1;  }  inline bool cmp\_eq(double x, double y){  return cmp(x, y) == 0;  }  inline bool cmp\_lt(double x, double y){  return cmp(x, y) < 0;  }  class Point {  public:  double x, y;  Point(double x\_ = 0.0, double y\_ = 0.0) : x(x\_), y(y\_) {} Point operator +(const Point &o) const {  return Point(x + o.x, y + o.y); }  Point operator -(const Point &o) const {  return Point(x - o.x, y - o.y); }  Point operator \*(const double &m) const {  return Point(m \* x, m \* y); }  Point operator /(const double &m) const {  return Point(x / m, y / m); }  // Producto punto  double operator \*(const Point &o) const {  return x \* o.x + y \* o.y; }  // Producto cruz  double operator ^(const Point &o) const {  return x \* o.y - y \* o.x; }  int cmp(Point o) const {  if (int t = ::cmp(x, o.x)) return t;  return ::cmp(y, o.y);  }  bool operator ==(const Point &o) const {  return cmp(o) == 0; }  bool operator !=(const Point &o) const {  return cmp(o) != 0; }  bool operator < (const Point &o) const {  return cmp(o) < 0; }  double Distance(const Point &o) const {  double d1 = x - o.x, d2 = y - o.y;  return sqrt(d1 \* d1 + d2 \* d2);  }  /\*Calcula la distancia entre el punto y la linea especificada por los dos puntos dados, is isSegment es verdadero, se tratan los dos puntos como un segmento  \*/  double Distance(const Point &p1, const Point &p2,  const bool &isSegment) const {  double dist = ((p2 - p1) ^ (\*this - p1)) / p2.Distance(p1); if (isSegment) {  double dot1 = (\*this - p2) \* (p2 - p1);  if (::cmp(dot1) > 0)  return sqrt((p2 - \*this) \* (p2 - \*this));  double dot2 = (\*this - p1) \* (p1 - p2);  if (::cmp(dot2) > 0)  return sqrt((p1 - \*this) \* (p1 - \*this));  }  return abs(dist); | distanceSq(Point2D pt)  boolean equals(Object obj)  abstract getX()  abstract getY()  int hashCode()  abstract void setLocation( x, y)  void setLocation(Point2D p)  ó  import java.io.\*;  public class Main {  public static class Point implements Comparable<Point> {  public static final double EPS = 1e-10;  public double x;  public double y;  public Point(double x, double y) {  this.x = x;  this.y = y;  }  public Point() {  this.x = 0.0;  this.y = 0.0;  }  public double dot(Point o) {  return this.x \* o.x + this.y \* o.y;  }  public double cross(Point o) {  return this.x \* o.y - this.y \* o.x;  }  public Point add(Point o) {  return new Point(this.x + o.x, this.y + o.y);  }  public Point subtract(Point o) {  return new Point(this.x - o.x, this.y - o.y);  }  public Point multiply(double m) {  return new Point(this.x \* m, this.y \* m);  }  public Point divide(double m) {  return new Point(this.x / m, this.y / m);  }  /\*Función para comparar dos números reales.( Se puede definir la precisión especificando el épsilon SIEMPRE se debe usar para comparar reales, nunca usar los operadores == o != ó <= , < , >= , > \*/  public static int cmp(double x, double y, double tol) {  return (x <= y + tol) ? (x + tol < y) ? -1 : 0 : 1;  }  public int hashcode()  {  return this.toString().hashCode();  }  public int compareTo(Point o) {  int t = cmp(x, o.x, EPS);  if (t != 0) return t;  return cmp(y, o.y, EPS);  }  public double Distance(Point o) {  double d1 = x - o.x, d2 = y - o.y; |
| --- | --- |

| }  friend ostream& operator <<(ostream &o, Point p) {  return o << "(" << p.x << ", " << p.y << ")";  }  };  int main()  {  Point a = Point(9,0);  cout << a << endl;  return 0;  } | return Math.sqrt(d1 \* d1 + d2 \* d2);  }  /\*Calcula la distancia entre el punto y la linea especificada por los dos puntos dados, is isSegment es verdadero, se tratan los dos puntos como un segmento  \*/  public double Distance(Point p1, Point p2, boolean isSeg) { double dist = (p2.subtract(p1)).cross(this.subtract(p1)) / (p2.Distance(p1));  if (isSeg) {  double dot1 = (this.subtract(p2)).dot(p2.subtract(p1)); if (cmp(dot1, 0.0, EPS) > 0) {  Point tmp = p2.subtract(this);  return Math.sqrt(tmp.dot(tmp));  }  double dot2 = (this.subtract(p1)).dot(p1.subtract(p2)); if (cmp(dot2, 0.0, EPS) > 0) {  Point tmp = p1.subtract(this);  return Math.sqrt(tmp.dot(tmp));  }  }  return Math.abs(dist);  }  public String toString() {  return "(" + this.x + "," + this.y + ")";  }  }  public static void main(String[] args)throws Exception  {  Point a = new Point(0,9);  System.out.println(a);  }  } |
| --- | --- |
| ----- | Line2D  Object clone()  boolean contains( x, y)  boolean contains( x, y, w, h)  boolean contains(Point2D p)  Rectangle getBounds()  abstract Point2D getP1()  abstract Point2D getP2()  abstract getX1()  abstract getX2()  abstract getY1()  abstract getY2()  boolean intersectsLine( x1, y1, x2, y2) boolean intersectsLine(Line2D l)  static boolean linesIntersect( x1, y1, x2, y2, |

|  | x3, y3, x4, y4)  ptLineDist( px, py)  static ptLineDist( x1, y1, x2, y2, px, py) ptLineDist(Point2D pt)  ptLineDistSq( px, py)  static ptLineDistSq( x1, y1, x2, y2, px, py) ptLineDistSq(Point2D pt)  ptSegDist( px, py)  static ptSegDist( x1, y1, x2, y2, px, py) ptSegDist(Point2D pt)  ptSegDistSq( px, py)  static ptSegDistSq( x1, y1, x2, y2, px, py) ptSegDistSq(Point2D pt)  int relativeCCW( px, py)  static int relativeCCW( x1, y1, x2, y2, px, py) to (x2,y2).  int relativeCCW(Point2D p)  abstract void setLine( x1, y1, x2, y2) void setLine(Line2D l)  void setLine(Point2D p1, Point2D p2) |
| --- | --- |
| ----- | Calendar-> GregorianCalendar  Calendar:  boolean after(Object when)  boolean before(Object when)  void clear()  void clear(int field)  int compareTo(Calendar anotherCalendar)  GregorianCalendar:  void add(int field, int amount)  Object clone()  boolean equals(Object obj)  void roll(int field, boolean up)  void roll(int field, int amount)  void setGregorianChange(Date date) |

| Lectura linea a linea:  #include<iostream>  #include<string>  #include<cstdio>  using namespace std;  int main(){  string line;  while( getline(cin,line) ){  cout << line << endl;  }  return 0;  } | Lectura linea a linea: Esta plantilla sirve para probar el programa con un arhivo en el directorio local y enviarlo para que lea de entrada estándar en los jueces online. Aunque algunoas hay que revisar que File no esté dentro de las clases prohibidas.  import java.io.\*;  public class MainPair{  private static final File \_ = new File("myArchivoLocal.txt"); public static void main(String[] args)throws Exception{  BufferedReader br = \_.exists()?  new BufferedReader(new FileReader(\_)):  new BufferedReader(new InputStreamReader(System.in)); String str;  while( (str = br.readLine()) != null ){  System.out.println(str);  }  }  } |
| --- | --- |
| Lectura Dato a Dato:  #include<iostream>  #include<string>  #include<cstdio>  using namespace std;  int main(){  string str;  while( cin >> str ){  cout << str << endl;  }  return 0;  } | Lectura Dato a Dato: Esta plantilla sirve para probar el programa con un arhivo en el directorio local y enviarlo para que lea de entrada estándar en los jueces online. Aunque algunoas hay que revisar que File no esté dentro de las clases prohibidas.  import java.io.\*;  import java.util.Scanner;  public class Main{  private static final File \_ = new File("myArchivoLocal.txt"); public static void main(String[] args)throws Exception{  Scanner sn = \_.exists()?  new Scanner(new FileReader(\_)):  new Scanner(new InputStreamReader(System.in));  String str;  while( sn.hasNext() ){  str = sn.next();  System.out.println(str);  }  }  } |
|  | Shape y Path2D  abstract void append(PathIterator pi, boolean connect)  void append(Shape s, boolean connect)  abstract Object clone()  void closePath() |

|  | boolean contains(x, y)  boolean contains(x, y, w, h)  static boolean contains(PathIterator ,x,y) static boolean contains(PathIterator ,x,y,w,h) static boolean contains(PathIterator ,Point2D p) static boolean contains(PathIterator ,Rectangle2D) boolean contains(Point2D p)  boolean contains(Rectangle2D r)  abstract void curveTo(x1, y1, x2, y2, x3, y3) Rectangle getBounds()  Point2D getCurrentPoint()  boolean intersects(x, y, w, h)  static boolean intersects(PathIterator,x,y,w,h) static boolean intersects(PathIterator,Rectangle2D) boolean intersects(Rectangle2D r)  abstract void lineTo(x, y)  abstract void moveTo(x, y)  abstract void quadTo(x1, y1, x2, y2)  void reset() |
| --- | --- |
| Plantilla C++ que funciona por lo menos en juez de la Uva con el problema Hashmat the brave warrior | Plantilla java que funciona por lo menos en juez de la Uva con el problema Hashmat the brave warrior  Ayudas Adicionales Java |