Programación de Ordenadores

Ingeniería Química

Curso 2007 - 2008

7

David Pelta

Depto de Ciencias de la Computación e I.A.

Universidad de Granada

Tipo de Datos String

- El tipo de datos *string* es uno de los tipos más utilizados por los programadores.
- Un *string* es una secuencia de caracteres, como por ejemplo *"Hola", "Química"*
- En C++, los strings se encierran entre comillas, las cuales NO son parte del string.
- El tipo string es parte del estandar de C++ y para poder utilizarlo debemos incluir el fichero de cabeceras "string" mediante la directiva

```
#include <string>
```

Declaración y Asignación de Variables de tipo *string*

- Para declarar variables de tipo string, haremos:

  string cadena1, cadena2;

- Para asignar valores a una variable, tenemos dos opciones:

  1. ***cadena1 = "mi primer string";***
  2. ***cin >> cadena1;***

- Para mostrar el valor de un string haremos :

  cout << cadena1

- Cuando se lee un string desde el teclado, unicamente una palabra se almacena en la variable. Si ante un:

  cin >> cadena1

  el usuario ingresa ***Ciudad de Granada***

  solamente ***"Ciudad"*** se almacenará en *cadena1.*

Si quisieramos almacenar mas de una palabra en una variable, debemos utilizar una "función" llamada *getline.*

getline(cin, cadena1);

Esta función, lee el valor de todas las teclas que se presionaron hasta el "Intro" y se almacenan en **cadena1**.

Si la cadena ingresada es "*Quimica Organica*", entonces diremos que *cadena1* tiene una longitud de 16 caracteres: las 15 letras + 1 espacio (también cuenta como un caracter).

Para saber la longitud de un string, podemos utilizar la función *length* de la siguiente manera:

int n = cadena1.length()

cout << cadena1.length()

Primero escribimos el nombre de la variable que nos interesa, luego un punto '.', y luego el nombre de la función.

Operaciones usuales con strings: Concatenación

Dos operaciones relevantes para la manipulación de strings o cadenas de caracteres son: la concatenación y la extracción de subcadenas.

La concatenacion es "pegar" dos cadenas para obtener una tercera.

Ejemplo:

```
string cad1 = "Jorge ";
string cad2 = "Borges";
string cad3 = cadena1 + cadena2;
```

La variable ***cad3*** contendrá "Jorge Borges".

Operaciones usuales con strings: Concatenación

Además se verifica que:

cad3.length() == cad1.length() + cad.length()

También podemos concatenar cualquier número de variables de tipo string y combinar con literales:

```
cad3 = cad1 + " Luis " + cad2;
```

La variable ***cad3*** contendrá "Jorge Luis Borges".

Extracción de subcadenas

Dada una cadena *S*, podemos obtener subcadenas (substrings) utilizando la función *substr().*

Así, la función *S.substr(inicio, longitud),* nos permitirá obtener la subcadena de S, que comienza en la posición *inicio* y tiene *longitud* caracteres.

Ejemplo:

```
string cad1 = "Programa de Hoy";
string cad2 = cad1.substr(0,3);
```

La variable ***cad2*** contendrá la subcadena ***"Pro"***. Al igual que ocurre con los vectores, la primera posición comienza en cero.

P	r	o	g	r	a	m	a		d	e		H	o	y
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

Operaciones Válidas

Comparaciones

Dadas dos cadenas *c1* y *c2*, podemos compararlas con los operadores de comparación usuales.

Igualdad: *c1 == c2.*
Menor que: *c1 < c2*
Mayor que: *c2 > c1*

Las comparaciones por menor / mayor utilizan el orden lexicográfico (similar al orden alfabético, pero incluyendo números y carácteres de control).

Strings como Vectores de Caracteres

Podemos asumir que un string es un vector de caracteres. Así, el string *S*, definido como:

string S = "HOLA"

se puede interpretar como un vector de 4 caracteres:

'H' , 'O' , 'L' , 'A'.

Los valores para variables de tipo *char*, se encierran entre comillas simples, mientras que los valores para *strings*, utilizan comillas dobles.

Por lo tanto 'H' ≠ "H"

Strings como Vectores de Caracteres

Viendo un string como un vector, podemos acceder directamente a cada letra, para observar o cambiar su valor:

string s = "Hello"
s[3] = 'p'
s[4] = '!'

Ahora el contenido de *s* es la cadena *"Help!"*

Naturalmente, el mismo resultado podemos obtenerlo utilizando funciones para strings:

s = s.substr(0,3) + "p!"

Tomamos la subcadena formada por las 3 primeras letras y luego concatenamos un valor literal de string.

Veamos un ejemplo que permite contar la cantidad de letras 'a' o 'A' que aparecen en una palabra.

Ejemplo: contar cuantas 'a' y 'A' aparecen

```
#include <iostream>
#include <ctime>
#include <string>

int main()
{string cadena = "Programacion de Ordenadores";
 int cant_letras, cant_a, i;

 cant_letras = cadena.length();
 cant_a = 0;
 for(i=0; i < cant_letras; i++)
   {if((cadena[i] == 'a')||(cadena[i] == 'A'))
        cant_a = cant_a + 1;
    }
cout << " El nro de letras A es : " << cant_a;
}
```

Más funciones de manipulación de strings

Si queremos analizar una cadena carácter a carácter, es mejor utilizar la notación de vectores.

Algunas funciones de manipulación de caracteres que pueden resultar de utilidad son (*h* es una variable de tipo *char)*:

isspace(h)	es *h* un espacio ?
isdigit(h)	es *h* un dígito ?
isalpha(h)	es *h* una letra ?
toupper(h)	transforma *h* a mayusculas
tolower(h)	transforma *h* a minusculas

Si queremos utilizar estas funciones debemos utilizar la siguiente directiva:

#include <cctype>

Ejemplo: pasar una cadena a mayusculas

```
#include <iostream>
#include <string>
#include <cctype>

using namespace std;
int main()
{string cadena = "programacion de Ordenadores";
 int i, cant_letras;
 cant_letras = cadena.length();
 for(i=0; i < cant_letras; i++)
   { cadena[i] = toupper(cadena[i])
    }
cout << " La cadena nueva es: " << cadena;
}
```

Ejercicios Adicionales

1. Implemente una función que indique si una palabra es un palíndrome (se lee igual de izquierda a derecha que de derecha a izquierda).
2. Implemente una función que reciba un string S y una letra X, y ponga en mayúsculas cada ocurrencia de X en S. (la función debe modificar la variable S).
3. Defina una función que reciba un string y lo muestre en orden inverso
4. Defina una función que reciba un string S y devuelva el mismo string S al cual se le han eliminado los espacios

Ejercicios Adicionales

1. La ruta nos aportó otro paso natural
2. Edit peptide
3. No solo no lo son
4. *Adivina ya te opina, ya ni miles origina, ya ni cetro me domina, ya ni monarcas, a repaso ni mulato carreta, acaso nicotina, ya ni cita vecino, anima cocina, pedazo gallina, cedazo terso nos retoza de canilla goza, de pánico camina, ónice vaticina, ya ni tocino saca, a terracota luminosa pera, sacra nómina y ánimo de mortecina, ya ni giros elimina, ya ni poeta, ya ni vida*

Estructuras en C++

- Una estructura es un tipo de dato definido por el usuario
- Se utiliza para agrupar un conjunto de valores relacionados
- Son un tipo compuesto de datos
- Contienen dos o mas miembros con tipo de dato pre-existente
- El tipo de dato de los miembros puede ser:
  - Un tipo de dato básico (int, double, char, …)
  - Vectores
  - Otra estructura

Declaración

Para declarar una estructura haremos lo siguiente

```
struct  tag_name
     {   data_type1 member1;
          data_type2 member2;
          …
          data_typen membern;
    };
```

struct, es una palabra reservada

Ejemplo

Declarar una estructura que permita almacenar una fecha con día, mes y año

```
struct  fecha
    {    int anio;
          int mes;
          int dia;
   };
```

Declaración y Uso de Variables

Para declarar variables de tipo fecha, haremos

struct fecha miFecha;

Dependiendo del compilador, puede ser suficiente indicar:

fecha miFecha;

Para acceder a las componentes de la estructura "fecha" utilizaremos el operador punto (.)

```
miFecha.dia = 12;
miFecha.mes = 9;
miFecha.anio = 2006;
```

Ejemplo

Declarar una estructura que permita almacenar el nombre y precio por kg. de un producto:

```
struct articulo
  { string nombre;
     double precio;
 };
```

Cont..

- Declarar una variable de este tipo:
  `articulo a1;`

- Almacenar “Doritos”, 1.50 en dicha variables:

```
a1.precio = 1.50;
a1.nombre = “Doritos”;
```

Es una variable de tipo double

Es una variable de tipo string

- Mostrar el contenido de la variable por pantalla:

```
cout << “Nombre:” << a1.nombre
     << “ Precio: ” << a1.precio << endl;
```

Vectores de Estructuras

Naturalmente, es posible declarar vectores de tipos definidos por el usuario.

Por ejemplo, podemos declarar un vector de 40 productos que se venden en una tienda:

```
articulo  tienda[40];
```

Guardar "Manzana" y 1.29 en la primer componente del vector:

```
tienda[0].precio = 1.29;
tienda[0].nombre = "Manzana";
```

Un simulador de bascula registradora

Queremos simular una báscula registradora usando la estructura "articulo".

Disponemos de un vector (una base de datos)

```
articulo productos[N];
```

El usuario ingresará códigos de artículo (podemos considerar que el código *i* representa el *i-esimo* producto) y peso, y el programa mostrará el nombre del artículo, el precio x kilo y el precio de la compra.

Al final, mostrará el total.

Paso de Estructuras como Parámetros a funciones

Escribir una función que permita mostrar los elementos de una variable de tipo “articulo”.

El prototipo de la función podría ser:

```
void  mostrar (articulo a);
```

mientras que la implementación sería:

```
void  mostrar (articulo a)
 {cout << “Nombre:” << a1.nombre
       << “ Precio: ” << a1.precio << endl;
 }
```

Vectores de Estructuras como parámetros a Funciones

Para pasar vectores de estructuras como parametros a funciones, procedemos de la manera que ya hemos visto:

```
void mostrarTodos (articulo P[], int n)
{int k;
 for (k = 0; k < n; k++)
  {cout << “Nombre:” << P[k].nombre
      << “ Precio: ” << P[k].precio << endl;
  }
}
```

Anidamiento de Estructuras

Un miembro de una estructura puede ser otra estructura. Por ejemplo, una biblioteca es un conjunto de secciones y cada seccion un conjunto de libros.

```
struct Libro{
  string titulo;
  string autor;
};
```

```
struct Seccion{
  Libro conjunto[MAX];
  int cantidad;
};
```

```
struct Biblioteca
{ Seccion secciones[MAX];
};
```

Como cargo/accedo a un libro particular?

Ejercicios

1. Defina una estructura que permita representar un punto en 2D.
   - Como haría para representar un punto en 3D.
   - y un punto en cualquier dimension
   - Como haría para representar un poligono? Muestre un ejemplo de uso

2. Sugiera una implementacion para el tipo de datos Conjunto de numeros enteros. Implemente las funciones que modelizen las operaciones de : Union, Pertenencia, Interseccion y Pertenencia.