Lenguaje C++
Introducción

- B. Stroustrup, mediados de los 80, AT&T

- Paso de la programación tradicional (C) a estilos de abstracción de datos y orientación a objetos
- Conserva características del C y añade nuevos conceptos

Lenguaje C++
Extensiones respecto al C

- No orientadas a objeto
- Orientadas a objetos

Lenguaje C++
Extensiones respecto al C

- No orientadas a objeto
  - Mejoras en entrada/salida - streams
  - Sobrecarga de funciones y operadores
  - Parámetros por defecto. Referencias

Lenguaje C++

```
#include<iostream.h>
        main()
        {//begin
                cout << “Hello world”;
        }//end
```

Lenguaje C++

- Organización de programas
  - archivos de declaraciones (.h)
    - Del sistema
    - Del usuario (#ifndef .... #define ..... #endif)
  - archivos de código fuente (.cc, .cpp)

Lenguaje C++

- Scope (alcance)
  - archivo: declaraciones que no pertenecen a función o clase alguna
  - Función: etiquetas (go to)
  - Local: cualquier nombre declarado en un bloque pertenece al bloque.
  - Clase: declaraciones asociadas a una clase
- Cada variable tiene un “scope” o un contexto

Lenguaje C++

- Operadores y sentencias
  - Sentencias
    - Misma sintaxis y semántica que las de C
  - Operadores idem sentencias excepto:
    - new, delete, delete[], ::
    - <<, >>
    - Sobrecarga

Lenguaje C++

- Declaración:
  - enumerados
    - enum status {error, ok};
  - apuntadores
    - int i = 25;
    - int *np;
    - np = &i;

Lenguaje C++

- Declaración:
  - referencias
    - int i = 5;
    - int& j = i;
    - i = 7;
    - cout << i << j; //cout : salida estandard
- Comentarios
  - /*.....*/, //.....

Lenguaje C++

- Entrada / salida
  - iostream.h, iostream
  - cout, cin, operator<<, operator>>
  - sobrecarga de los operadores para entrada salida de tipos definidos por el usuario.

Lenguaje C++

- Variables y aritmética.
- Todo nombre y/o expresión tiene un tipo.
- El tipo determina las operaciones que se pueden realizar con el.
- Ejemplo:

  int dato;

  operaciones: + - / *

Lenguaje C++

- Tipos fundamentales.

char	float
short	double
int	long double
long	(unsigned)

- Operaciones asociadas.
  - +,-,*,/,%, ==,!=,<,>,<=,>=,

Lenguaje C++

- Tipos fundamentales.
  - Conversiones implícitas.
  - En asignaciones y operaciones algebraicas el lenguaje lleva a cabo todas las conversiones implícitas que faciliten el uso y sean seguras.

Lenguaje C++

- apuntadores y vectores.

```
char v[10]; //vector de 10 caracteres (0 a 9)
char *p; // apuntador a un carácter
p = &v[3]; //p apunta al cuarto elemento de v
```

Lenguaje C++
Funciones

- Paso de argumentos (siempre es por valor)
  - Por valor - seguridad
    - copiado en el almacenamiento local de la función
    - la función accede a la copia
  - Por referencia - velocidad - retornos múltiples
    - se copia la dirección del objeto que se esta pasando
    - la función accede al objeto
  - Por referencia constante - seguridad + velocidad
- Excepción: vectores: paso por referencia

Lenguaje C++
Módulos

- El concepto de módulo en c++ se refiere a unidades de compilación separadas.
- Similar al concepto en C.

Lenguaje C++
Tipos Derivados

- * apuntador - relación con vectores
- & referencia
- [] vector
- () función
- struct {} estructuras - union {}- fields.
- void (se comporta como fundamental pero solo se lo utiliza como parte de derivados)

Lenguaje C++
Tipos Derivados

- (tipo) expresión: especifica conversión
- tipo(expresión): especifica conversión
    - float r = float(1); // explicito tipo constructor
    - char * p = (char *) 0777; //como en C si el nombre
      //no es simple o typedef
- apuntadores - operaciones asociadas
  - int *p; //apuntador a entero
  - int **p; //apuntador a apuntador a entero
  - int (*vp) [10]; //apuntador a vector de 10 enteros
  - int (*fp) (char, char *);// apuntador a función

Lenguaje C++
Estructuras

- Conjunto de elementos de distinto tipo.
- Inicialización. Constructores.
- Operaciones asociadas (==, <)
- Equivalencia de tipo.
  - struct s1 { int a; }; != struct s2 {int a;};
  - El tipo struct es distinto a cualquier tipo fundamental

Lenguaje C++
Referencias

- Nombre alternativo para un objeto.
- Uso principal
  - Argumentos y retorno de funciones y operadores.
- Notación: X& => referencia a X.

Lenguaje C++
Soporte para abstracción de datos

- Definición de un conjunto de operaciones asociadas a un tipo.
- Restricción del conjunto de operaciones que se pueden realizar sobre un tipo a las definidas para el mismo.
- CLASE
  - nombre
  - datos miembros
  - funciones miembro
  - control de acceso

Lenguaje C++
Soporte para abstracción de datos

- Inicialización y destrucción.
- Inicialización y asignación.
- Patrones.
- Gestión de excepciones.
- Conversiones de tipo

  complex a = complex(1);
- Múltiples implementaciones

Lenguaje C++
Soporte para orientacion a objetos

- Funciones virtuales. Polimorfismo.
- Clases abstractas.
- Clases patrón.
- Herencia simple y múltiple.
- Encapsulado.

Lenguaje C++

- Bloques {}
- Operador ::

```
int x;
void f()
{
    int x = 1;
    ::x = 2;
}
```

Lenguaje C++

- Los objetos se crean al llegar a su definición.
- Los objetos se destruyen cuando salen del bloque.
- Los objetos globales se crean una sola vez y se destruyen al terminar el programa.
- Los objetos static son similares a los globales.

Lenguaje C++

- Almacenamiento de objetos
  - estático (asignada al iniciarse el programa)
  - automático (asociada al bloque)
  - en la memoria disponible (obtenida con new)

Lenguaje C++

- Enunciados
  - if, if else,
  - switch case default
  - for while do while
  - break continue return goto

Lenguaje C++

Clases.

Tipo abstracto de datos definido por el usuario

Interface e implementación

```
class X {
    private:
        int i;
    public:
        f();
    protected:
        g();
    };
```

```
struct X {
    private:
        int i;
    public:
        f();
    protected:
        g();
    };
```

```
X *const this;
```

Lenguaje C++
clases

- Función miembro const
  - El tipo this de la función será const X * const
- Función miembro inline
- Función miembro virtual
- Función miembro virtual pura

Lenguaje C++
clases

- Constructores
  - Funciones con el mismo nombre de la clase pero que no retornan nada. - Excepciones.
  - Constructor de copia
  - Constructor por defecto
  - Constructores como conversores de tipo
- Destructores
  - virtuales

Lenguaje C++
clases

- Friend
- Calificación de nombres
  - operador ::
- Clases anidadas
- Miembros estáticos
- apuntadores a miembros
  - typedef void (cl::*PFM)(int)
  - PFM pf = &cl::f;

Lenguaje C++
clases

- Construcción de objetos
  - automáticos
    - variables locales: se ejecuta su constructor cada vez que el programa pasa por la declaración
      - secuencia de construcción - destrucción
    - inicialización y asignación
  - estáticos
    - utilizada para asegurar la correcta inicialización y destrucción de objetos de bibliotecas : cin, cout
  - en memoria disponible
    - new T, delete

Lenguaje C++
clases

- Construcción de objetos
  - como miembros de otro objeto
    - sintaxis de inicialización :
    - secuencia de llamada a los constructores 1 los miembros
    - secuencia de llamada a los destructores 1 el objeto
    - uso de apuntadors a objetos miembros
  - vectores de objetos
    - los objetos deben tener constructor por defecto
    - la destrucción de objetos es implícita si el vector es automático
    - la destrucción es explicita cuando se trata de apuntadors
      - delete, delete[]

Lenguaje C++
clases y herencia

- Construcción de objetos
  - gestión de memoria espécifica
    - sobrecarga de new y delete
- Clases derivadas
- Herencia
  - publica, protegida y privada. Datos, funciones, tipos
  - conversión de tipo
  - class x : public y {};
  - construcción - destrucción

Lenguaje C++
clases y herencia

- Jerarquías
  - funciones virtuales
  - clases abstractas
- Herencia múltiple
  - class a :public b, public c {};
  - base múltiple
  - base virtual
- Destructores virtuales - size_t de delete

Lenguaje C++
clases y herencia

- Constructores virtuales
  - no pueden serlo
  - se los puede simular
    - típicos: por defecto y de copia
-

```
class X {
public:
        X();
virtual X* NewX ()
    { return new X(); }
};
void f(X * p1, X *p2)
{ X *p3 = p1->NewX(); X *p4 = p2->NewX(); }
```

```
class Y : public X {
    public:
        Y();
     X* NewX ()
    {return new Y(); }
};
```