TDA
Tipo de Dato Abstracto

UNLA – Licenciatura en Sistemas
Algoritmos y Estructuras de Datos

Lic. Alejandro Sasin
Diego Cañete

Agenda

- Repaso
- Introducción a TDA
- Componentes de un TDA
- Ejemplo
- Características Deseables

Objetivo

- El objetivo de esta clase es que el alumno comprenda el concepto de TDA y su aplicación como paradigma de programación.

Repaso

Repaso

- Qué era un archivo .h? Para qué lo usamos?
- Qué hace el precompilador?
- El compilador?
- Y el linker?
- Para qué se usa #define, #include?
- Qué es y cómo se resuelve las inclusiones circulares?

Introducción

Introducción

- Qué es un paradigma?

  conjunto de creencias, suposiciones, leyes, métodos o técnicas sobre las que un grupo se basa para realizar cierta actividad.

  - Paradigma educativo
  - Paradigma científico
  - Paradigma de programación

Introducción

Introducción

- Paradigma de la programación estructurada:
  - Ejecución Secuencial.
  - Instrucciones condicionales.
  - Instrucciones iterativas o bucles.
- Un paso adelante: programación funcional (interacciones entre funciones).
- Un paso adelante: TDA (interacciones entre entidades independientes).
- Un paso adelante: Programación Orientada a Objetos.

Qué estrategia es mejor?

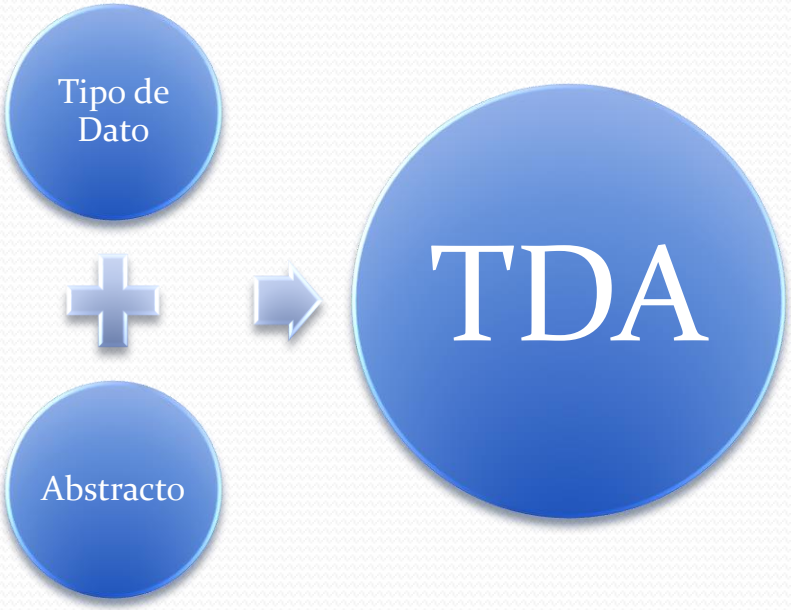
- Tengo que representar a un empleado.

a) Uso variables sueltas y funciones que las reciben y modifican.
b) Defino un tipo TEmpleado con un struct que junte todas las variables. Las funciones manejan directamente el TEmpleado.
c) Uso TDA.

Vamos a ver que... la mejor opción es: TDA.

Qué es un TDA?

- Tipo de Dato Abstracto:
  - Un TDA es una abstracción matemática que encapsula una colección de atributos accesibles solo mediante un conjunto de operaciones válidas.



Tipo de Dato

- A qué nos referimos cuando hablamos de un tipo?
  - A un conjunto de objetos con ciertas características en común.

- Qué define un tipo:
  - El rango de valores que puede tomar un dato del tipo.
  - El conjunto de operaciones permitidas sobre datos del tipo.

Tipo de Dato

- Qué es una instancia?
  - Un dato es una instancia de un tipo.

- No hay que confundir int con un dato; si decimos

  ```
  int num;
  ```

  num es una instancia del tipo int y sí constituye un dato, int es sólo el tipo al que pertenece.

Abstracción

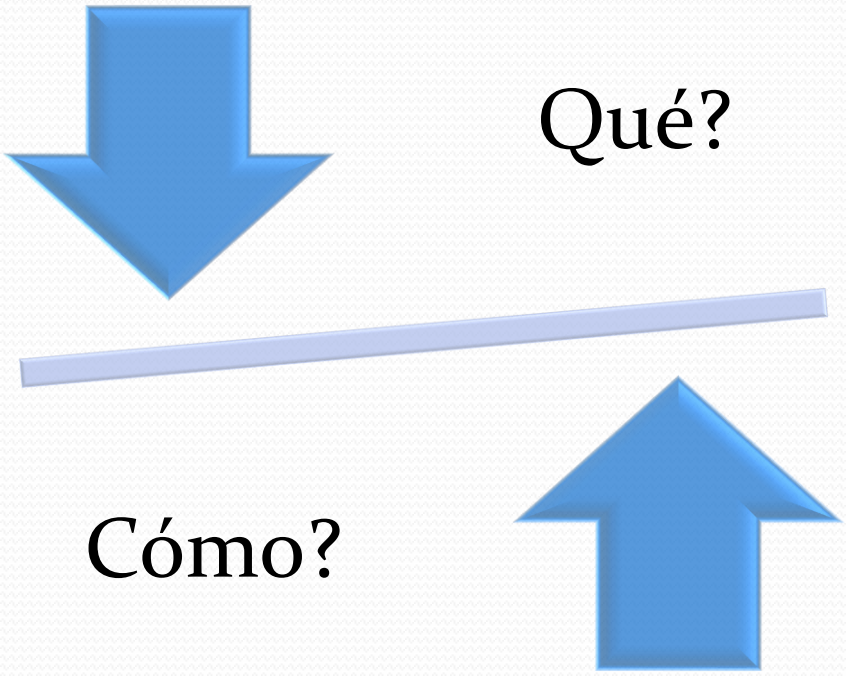
- Abstraer consiste en clasificar los atributos de la entidad en estudio y descartar aquellos que resultan irrelevantes.

- La abstracción resulta una herramienta de vital importancia ya que sin ella, no podríamos usar una computadora, estudiar, jugar, conversar…

Abstracción

- Cómo manejamos?

Interfaz vs Implementación



- Qué me interesa de un int, un char, o de la función printf?

Entonces…

- TDA nos va a permitir analizar los elementos que componen nuestra aplicación, desarrollarlos y combinarlos para resolver el problema en cuestión.

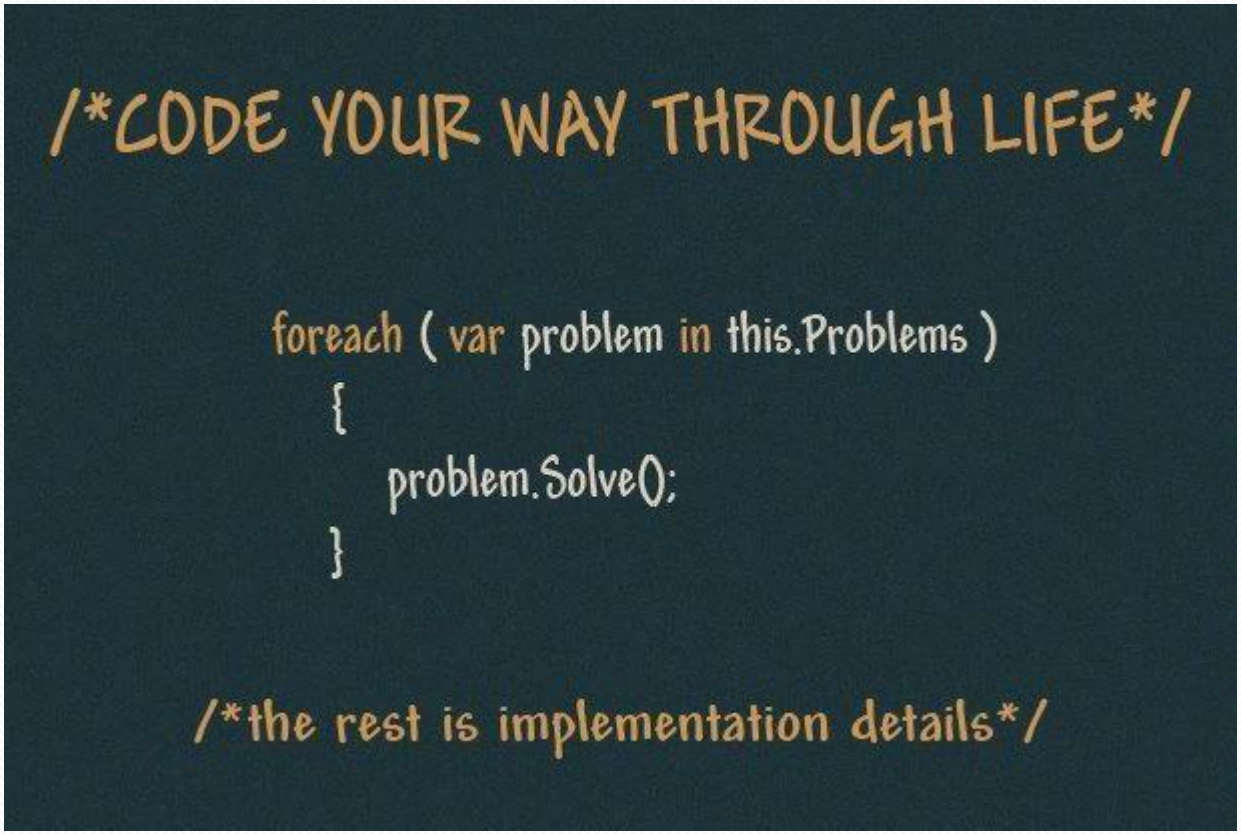
- Implica un cambio en la manera de pensar. Un cambio en la estrategia de resolución de los problemas. Un cambio de paradigma.

Para qué?

- Para qué queremos utilizar TDAs en la construcción de software?
  - Mejora la claridad del software.
  - Aumenta la elegancia del diseño.
  - Disminuye el impacto de los cambios.
  - Ayuda a la comunicación.
  - Aumenta el nivel de abstracción de los componentes.
  - Disminuye el costo de mantenimiento.
  - Potencia la reutilización de código.
  - Los programas se piensan como TDAs comunicándose entre sí. Se piensa en Entidades del Negocio.

Apuntamos a…

- Lograr…



Componentes de un TDA

Componentes de un TDA

- Un TDA se compone de:
  - Tipo de Dato
  - Axiomas
  - Operaciones
  - PreCondiciones y PostCondiciones

Componentes (Tipo de Dato)

- Representación de un tipo.
- Los datos de un TDA se llaman atributos.
- Características:
  - No deben regirse estrictamente por la propiedades físicas de los datos (si quiero desarrollar un TDA Nro Teléfono, no necesariamente debo usar un número).
  - Los tipos estructurados (compuestos), deben manejarse en forma independiente de los tipos de sus elementos (Concepto de Genericidad, ej. Vector[T]).
  - No necesariamente todos los valores posibles para el tipo de dato base lo serán para el TDA.

Componentes (Axiomas)

- Los axiomas son proposiciones lógicas que deben cumplirse siempre. Son aceptadas sin requerir una demostración previa.
- Ejemplo: Axiomas de Peano:
  - El 1 es un número natural.
  - Si *n* es un número natural, entonces el sucesor de *n* también es un número natural.
  - El 1 no es el sucesor de ningún número natural.
  - Si hay dos números naturales *n* y *m* con el mismo sucesor, entonces *n* y *m* son el mismo número natural.
  - Si el 1 pertenece a un conjunto, y dado un número natural cualquiera, el sucesor de ese número también pertenece a ese conjunto, entonces todos los números naturales pertenecen a ese conjunto.

Componentes (Operaciones)

- Las operaciones son también llamadas primitivas y es el único vínculo entre el usuario y el TDA. No existe ninguna otra forma de interactuar con ellos.
- Debe ser un conjunto tal que permita el correcto desempeño del TDA para el contexto en el que fue concebido.
- La cantidad de operaciones debe ser lo menor posible para disminuir la “superficie de contacto”, es decir los puntos de dependencia del TDA.
- Siempre el primer parámetro de una operación es una instancia del TDA.

Componentes (Operaciones)

- Operaciones obligatorias
  - Constructor
    - Es la primitiva que reserva recursos, inicializa valores y permite que pueda ser utilizado un TDA.
  - Destructor
    - Es la primitiva que libera todos los recursos tomados por un TDA. Luego de su ejecución, el TDA no podrá ser utilizado.

Componentes (Operaciones)

- Operaciones obligatorias
  - Getters para atributos
    - Un get es una primitiva utilizada para obtener el valor de un atributo del TDA.
    - La sintáxis es:
      - getNombreAtributo();
  - Setters para atributos.
    - Un set, es la primitiva utilizada para asignar un valor a un atributo del TDA.
      - setNombreAtributo();

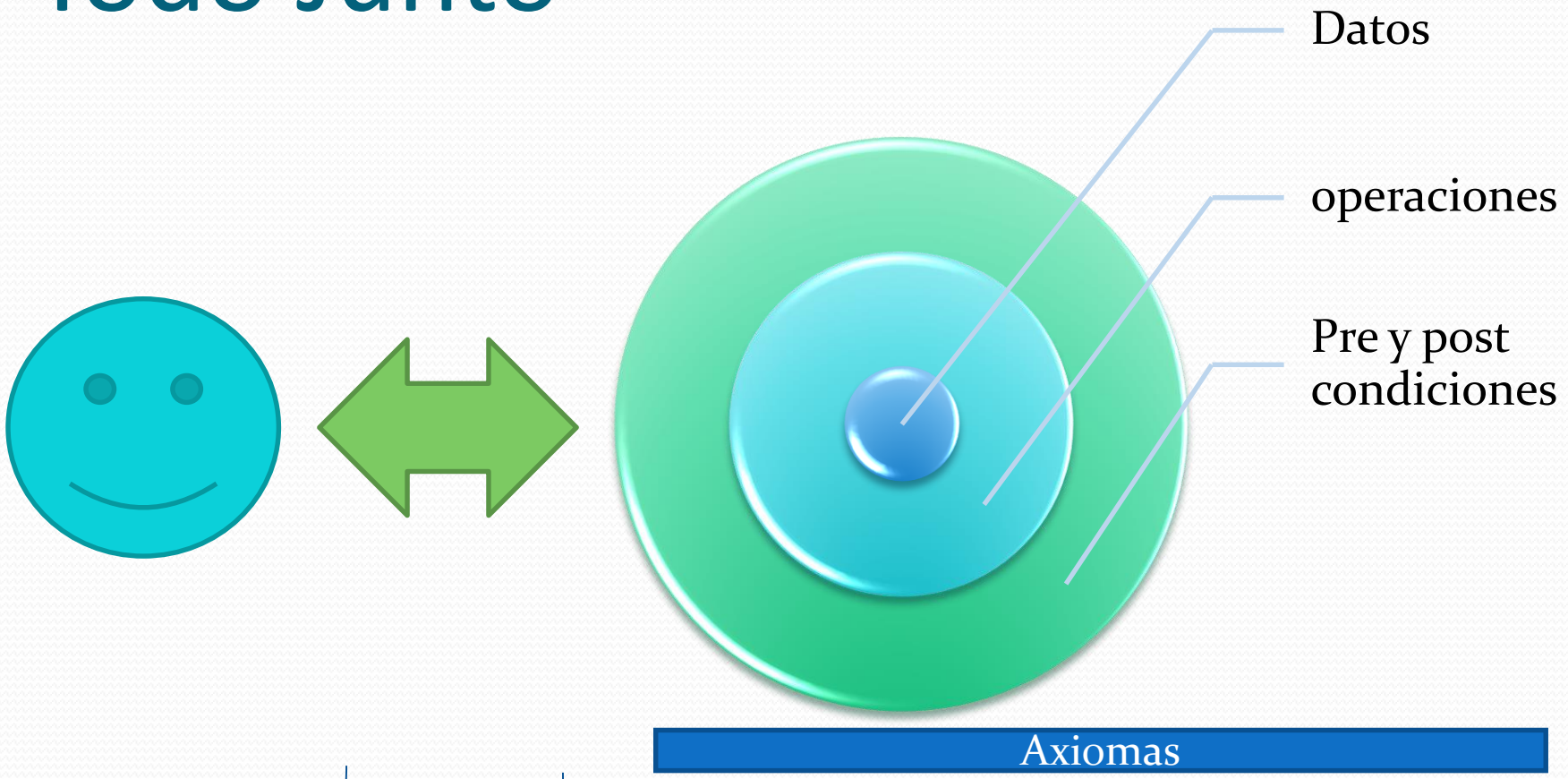
Componentes (Pre y Post)

- Las precondiciones son proposiciones lógicas que deben cumplirse antes de la ejecución de de una primitiva. Reflejan la realidad de que no todos los estados posibles son válidos para el TDA (no puedo pedir el primer carácter a un String vacío).
- Las postcondiciones son proposiciones lógicas que resultan verdaderas luego de la ejecución de una primitiva.
- Programación por contrato

Componentes (Programación por contrato)

- Se crea un contrato entre quién construye el TDA y quién lo utiliza mediante el cual se especifica el comportamiento del TDA y todas las operaciones para interactuar con él.
- El contrato debe cubrir todos aquellos aspectos que formen parte de la abstracción que se realizó, es decir aquellos que den soporte al nivel de detalle con el que se está estudiando el objeto en cuestión.
- El contrato es la herramienta que permite separar efectivamente la interfaz de la implementación. El qué del cómo.
- Permite que quien construya el TDA modifique su implementación según le parezca conveniente sin que esto impacte en el comportamiento esperado por el usuario.

Todo Junto


Datos
operaciones
Pre y post condiciones
Axiomas
USUARIO
TDA

Ejemplo

Ejemplo

- A modo de ejemplo, definiremos el TDA “radio digital”. Una radio digital es un dispositivo que permite sintonizar estaciones de radio mediante un selector de frecuencias digital.

Ejemplo Radio Digital

- Los atributos de la radio serán:
  - Estado: Prendida o Apagada.
  - Frecuencia: Frecuencia de la radio que se quiere escuchar.
  - Banda: AM o FM.
  - Volumen: volumen con el que se escuchará la radio.

Ejemplo Radio Digital

- Los axiomas del TDA son:
  - 0 <= Volumen <= 10
  - 550 <= FAM <= 1700
  - 95 <= FFM <= 108
  - FAM = 10 * X + 550
  - FFM = 0.1 * X + 95

X es un entero.

Ejemplo Radio Digital

- Las operaciones que podremos realizar son:
  - Encender y Apagar.
  - Subir y Bajar el volumen.
  - Subir y Bajar la frecuencia.
  - Cambiar la Banda.
  - Etc

Ejemplo Radio Digital

- Operaciones - Encender:
  - QUE:
    - Prepara el equipo para poder escuchar la radio.
  - COMO:
    - La corriente ingresa por el enchufe y llega al circuito….
  - PRE:
    - La radio se encuentra enchufada a la corriente.
  - POST:
    - El equipo comienza a transmitir en la radio X por defecto.

Ejemplo Radio Digital

- Operaciones - Apagar:
  - QUE:
    - Deja el equipo apagado, sin emitir sonido y sin consumir energía.
  - COMO:
    - Se corta la corriente del circuito y…
  - PRE:
    - El equipo ha sido prendido con la operación “encender”.
  - POST:
    - El equipo deja de transmitir y ya no consume más energía.

Ejemplo Radio Digital

- Subir Volumen:
  - QUE:
    - Aumenta la intensidad del sonido.
  - COMO:
    - Existe una resistencia que varía con….
  - PRE:
    - La radio ha sido prendida con la operación “encender”.
  - POST:
    - Se incrementa el volumen de la radio, teniendo como tope el volumen máximo.

Ejemplo Radio Digital

- Estructura de datos
  - Los TDAs contendrán un conjunto de datos para dar soporte a su comportamiento. Estos datos deberán permanecer ocultos para el usuario, protegidos por un conjunto de primitivas.
- En el caso del TDA Radio, algunos atributos podrían ser:
  - Frecuencia AM, frecuencia FM.
  - Banda seleccionada.
  - Estado.
  - Volumen.
- En c++ este conjunto de datos es definido dentro de un struct. Siempre se utiliza un struct, aún cuando se trate de un solo dato. De esta forma, se simplifica el agregado de nuevos atributos sin modificaciones en el código.

Características Deseables de un TDA

Características Deseables de un TDA

- Correcto
  - Un TDA es funcionalmente correcto si **se comporta de acuerdo a la especificación** de las funciones. La corrección es una propiedad matemática que establece una equivalencia entre el TDA y su especificación.
  - La corrección es un **cualidad absoluta**, cualquier desvío de la especificación hace que el sistema sea incorrecto.

- Confiable
  - En términos generales, un TDA es confiable **si el usuario puede contar con él**. La confiabilidad se mide con **estudios estadísticos**, es decir, la probabilidad de que el TDA funcione correctamente en un intervalo específico.
  - La confiabilidad es relativa, si la consecuencia de un error no es grave, el sistema puede aún ser confiable.

Características Deseables de un TDA

- Robusto
  - Un TDA es robusto si se comporta "razonablemente" aún para situación que no fueron previstas durante su construcción.
  - La robustez es difícil de definir puesto que si pudiéramos especificar completamente como hacer que un TDA responda "razonablemente" ante cualquier situación, la robustez se transformaría en corrección.
  - La robustez se relaciona con la especificación de las precondiciones. En ocasiones, el usuario puede llamar a una primitiva sin cumplir con las precondiciones. No es necesario verificar que se cumplan las condiciones impuestas, pero hacerlo brindará robustez al TDA.

- Eficiente
  - El rendimiento de un TDA se mide en función de la correcta utilización de los recursos.

Características Deseables de un TDA

- Amigable
  - El TDA es amigable si los usuarios lo encuentran fácil de utilizar. Como vemos, es una característica subjetiva.
- Verificable
  - Un TDA es verificable si es fácil comprobar sus características y propiedades, establecidas en la especificación.
- Portable
  - Un TDA es portable si se puede utilizar en distintos entornos, por ejemplo, distintos SO.

Características Deseables de un TDA

- Evolutivo
  - Se busca que el TDA evolucione ante cambios en su medio ambiente para seguir cumpliendo con su especificación. Se modifica el producto para obtener una nueva versión que cumpla con nuevas especificaciones.
- Reutilizable
  - Se pretende que un TDA pueda ser utilizado sin modificaciones en varios sistemas de software, incluso cuando el TDA se haya construido para una aplicación y se pretenda reutilizar en otra mucho tiempo después.
- Reparable
  - Un TDA es reparable si puede ser corregido con una cantidad de trabajo y recursos limitada. Un TDA bien diseñado es fácil de reparar.

Características Deseables de un TDA

- Comprensible
  - Un TDA debe ser fácil de entender, tanto para el usuario como para el constructor.
- Interoperable
  - Un TDA es interoperable si puede trabajar en conjunto con otros TDAs. Un TDA que solo puede trabajar aislado, no será utilizado en la construcción de un sistema. Es mediante la relación con otros TDAs que se logra sacar el máximo provecho de la funcionalidad construida.
- Productivo
  - El TDA es productivo si genera resultados de valor para un usuario.
- Oportuno
  - Un TDA es oportuno si entrega los resultados en el tiempo especificado.

Características Deseables de un TDA

- "Mantenible"
  - Un TDA es "mantenible" si es fácil realizar modificaciones luego de su versión inicial. Existen tres tipos de mantenimientos:
    - Correctivo: Corrige errores que permanecen luego de ser liberada la versión del TDA.
    - Adaptativo: Realiza modificaciones para adaptar el TDA a cambios en la especificación.
    - Perfectivo: Realiza modificaciones para perfeccionar el funcionamiento del TDA.

Fin


Gracias
Merci
Hvala
Ngiyabonga
Xié
Barka
Moltes
Dankon
Maraba
Maketai
Paldies
rhât
olun
Grazzie
Ntyox
Grazias
Bedankt
Thanks
Tänan
Mantiox
Murakoze
Tack
qúi
pai
leibh
Kaigai
dimo
Kommol
Shokrán
Dannaba
Mwebare
Dekoju
Khawp
Arigato
Emitekati
Tesekkür
Trugarez
Ashoge
Merçì
Tsin'aen
Matóndo
Fa'afetai
gràcies
agaibh
Sag
jai
Dakujem
so
Tinotenda
Spaisíva
ek
Multumesc
Marahaba
Aalghïstapcham
Syaabaas
GyalailaaThai
Takk
Tak
Blagodaram
òn
Faleminderit
Shukuriyyaa
chawe
magah
Yuspagara
Matu
Dyakooyu
Ngeyabonga
Kili
Dyuspagrasunki
Xie
Evgaristó
maluhlap
Mahalo
Shterakravetsun
Gunasakulila
TashakkurBulgaro
Tapaidh
Webale
Go
Rakhmat
Khrap
Obrigado
Imela
Gmadlob
Alla
maith
suksama
Eskerrik
Fafetai
Dêkuji
Ha'evete
Uzbezco
Rahmet
Danke
mamexes
Gratias
Netjer
Dios
raibh
asko
Kiitos
blu
Puno
todâ
Ah
hvala
Ashi