



PROGRAMACION

AVANZADA

ING. INDUSTRIAL

Profesora: Cristina Delia Cruz



ESTRUCTURA DE UN PROGRAMA EN C++

```
1 #include <iostream>
2
3 #include <stdlib>
4
5
6 [Declaración de Variables Globales]
7
8 int main()
9 {
10     [Declaración de Variables Locales]
11
12
13
14
15
16 }
```

Instrucciones Declarativas

Función Principal

Inicio del Programa

Cuerpo del Programa

Fin del Programa



ESTRUCTURA DE UN PROGRAMA EN C++

#include <Libreria_Solicitada>

Con esta directiva de preprocesador se incorporan las bibliotecas de funciones que se van a utilizar

Las librerías más utilizadas son:

- **<iostream>** contiene las funciones para ingresar y mostrar datos.
- **<math.h>** contiene las funciones matemáticas comunes.
- **<time.h>** contiene las funciones para tratamiento y conversión entre formatos de fecha y hora.
- **<stdio.h>** contiene los prototipos de las funciones, macros y tipos para manipular datos de E/S.
- **<stdlib.h>** contiene tipos, macros y funciones para la conversión numérica, generación de memoria y tareas similares.
- **<string.h>** contiene los prototipos de las funciones y macros de clasificación de caracteres.



ESTRUCTURA DE UN PROGRAMA EN C++

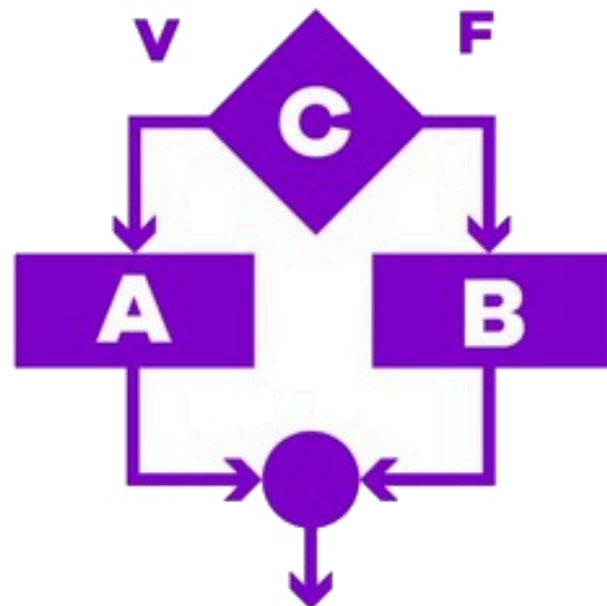
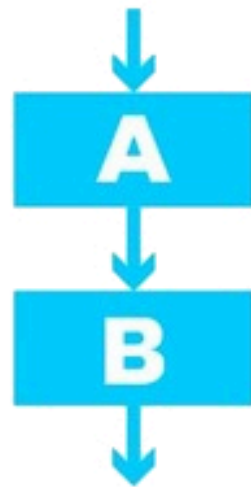
Setencia

Definen la lógica de un programa o subprograma

Manipulan los datos para producir el resultado deseado por el usuario del programa

En programación estructurada, se tendrán sentencias que permiten:

- Estructura secuencial
- Estructura selectiva
- Estructura repetitiva



Dato

Un dato es cualquier valor o pieza de información que un programa puede utilizar.

Estos datos pueden ser números, texto, fechas, imágenes, etc.

Para que la computadora pueda trabajar con estos datos, es necesario definir su tipo y cómo se representarán en la memoria.

Los datos en un programa se pueden representar a través de:

- Variables
- Constantes



```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int a;
8      float b = 4.7;
9      char c = 'k';
10     bool d = true;
11     return 0;
12 }
```




TIPOS DE DATOS EN C++

DATOS SIMPLES EN C++

C++ es un lenguaje de programación que hereda muchos conceptos del lenguaje C, es un lenguaje compilado y fuertemente tipado, lo que significa que en las variables con las que trabajamos hay que indicar el **tipo del dato** que van a guardar cuando se declaran, lo que puede hacer que tengamos problemas y se generen errores.



TIPOS DE DATOS EN C++

Nombre	Descripción	Tamaño*	Rango de valores*
char	Carácter o entero pequeño	1byte	con signo: -128 to 127 sin signo: 0 a 255
short int (short)	Entero corto	2bytes	con signo: -32768 a 32767 sin signo: 0 a 65535
int	Entero	4bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
long int (long)	Entero largo	8bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
bool	Valor booleano. Puede tomar dos valores: verdadero o falso	1byte	true o false
float	Número de punto flotante	4bytes	3.4e +/- 38 (7 digitos)
double	De punto flotante de doble precisión	8bytes	1.7e +/- 308 (15 digitos)
long double	Long de punto flotante de doble precisión	8bytes	1.7e +/- 308 (15 digitos)



DECLARACION DE VARIABLE EN C++

Es preciso declarar las variables antes de utilizarlas.

La sintaxis de declaración es:

`tipo nombre_de_variable[=valor];`

Lo que está entre corchetes es opcional, se puede o no dar el valor a la variable al mismo tiempo que se la declara. Además, se puede declarar varias variables del mismo tipo separándolas con comas. Si se quisiera declarar variables como las básicas, sería así:

1. `int T=1401, variable=1, a;`
2. `float pi=3.14, decimal, otra=0.23;`
3. `char letra='C', mas;`
4. `bool dicho=true, bandera=false;`

Nota: después de declarar una serie de variables de un mismo tipo hay que poner un ;



ENTRADA Y SALIDA EN C++

Las bibliotecas estándar de C++ proporcionan un amplio conjunto de capacidades de Entrada/Salida (E/S). C++ utiliza E/S a prueba de tipos. Cada operación de E/S se realiza automáticamente en una forma sensible con respecto al tipo de datos.

cin y **cout**: **cin** es el flujo de entrada estándar que normalmente es el teclado y **cout** es el flujo de salida estándar que por lo general es la pantalla. Para utilizar cualquiera de estos flujos debe incluirse **iostream** que es el archivo de encabezado del flujo de **entrada/salida**.



E/S DE DATOS EN C++

En **C++** se puede usar el flujo **cout** para imprimir información en la pantalla, utilizando el operador de salida de flujo "**<<**". Se puede **mostrar**: una cadena de texto, el contenido de una variable, el resultado de una expresión aritmética, etc. La constante **endl** imprime un retorno de carro. También puede utilizarse "**\n**".

Ejemplos:

```
cout<<"¡Hola Mundo...!";
```

```
cout<<"El valor de x es "<<endl;
```

El flujo de entrada **cin** permite introducir datos desde el teclado y almacenarlos en variables previamente declaradas.

Ejemplo:

```
int main() { int x;
```

```
cout<< "Ingrese un numero entero: ";
```

```
cin >> x;
```

```
cout<< "El numero que ingresaste es: ";
```

```
cout<< x << endl; }
```

Codificación

```
1  #include <iostream>
2  using namespace std;
3
4  main() {
5      int numero;
6      cout<<"Escriba Un Numero: "<<endl;
7      cin>>numero;
8      cout<<"El numero Insertado es: "<<numero<<endl;
9      return 0;
10 }
11
```

Ejecución

```
C:\Program Files (x86)\Zinjal\ X + v
Escriba Un Numero:
458
El numero Insertado es: 458
```

```
<< El programa ha finalizado: codigo de salida: 0 >>
<< Presione enter para cerrar esta ventana >>
```



ASIGNACIÓN DE VARIABLES Y EXPRESIONES ARITMÉTICAS

Para asignar un valor a una variable se utiliza el operador "=".

Es posible asignar valores al momento de declarar las variables;

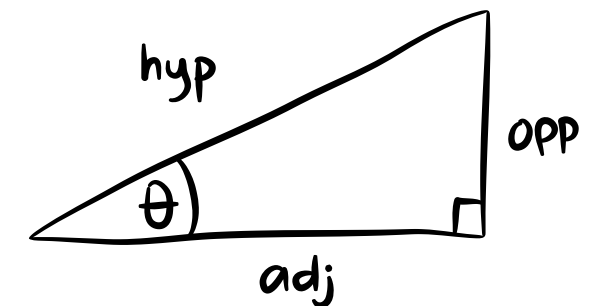
```
5 int var1 = 1;
6 char var2 = 'a';
7 float var3 = 13.5;
8 double var4 = 11.3;
9 bool var5 = true;
10
11 return 0;
```

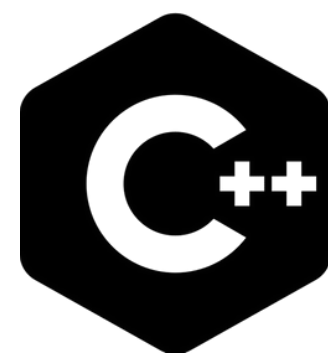
Una **operación aritmética** es un proceso matemático básico que involucra números y operadores aritméticos. Las operaciones aritméticas forman la base de las matemáticas y son fundamentales en la programación, así como en la resolución de problemas numéricos.

$$X = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$y = mx + b$$

$$ax + by = C \quad \sin(\theta) = \frac{\text{opp}}{\text{hyp}}$$





OPERADORES

ARITMETICOS EN C++

Tipo	Operadores
Asignación	=
Aritmético	Potencia: <i>powf(x,y)</i> (librería <i>math.h</i>); <i>x</i> , <i>y</i> valores numéricos Producto: * Cociente: / Módulo o resto: % Sumar: + Diferencia: -
Alfanuméricos	Operaciones con cadenas (librería <i>string.h</i>) <i>strcat(s,t)</i> ; concatena <i>t</i> al final de <i>s</i> . <i>strcmp(s,t)</i> ; compara <i>s</i> y <i>t</i> , retornando negativo, cero, o positivo para: <i>s</i> < <i>t</i> , <i>s</i> == <i>t</i> , <i>s</i> > <i>t</i> . <i>strcpy(s,t)</i> ; copia <i>t</i> en <i>s</i> . <i>strlen(s)</i> ; retorna la longitud de <i>s</i> . donde <i>s</i> y <i>t</i> son variables de tipo cadena.
Lógicos	Negación (NO, NOT): ! Conjunción (Y, AND): && Disyunción (O, OR):
Relacionales	Igual: == Distinto: != Mayor: > Mayor o igual: >= Menor: < Menor o igual: <=



OPERADORES ARITMÉTICOS

En C/C++ se pueden formar expresiones aritméticas con los operadores comunes: **+** (suma), **-** (resta), ***** (multiplicación), **/** (división), **%** (resto)

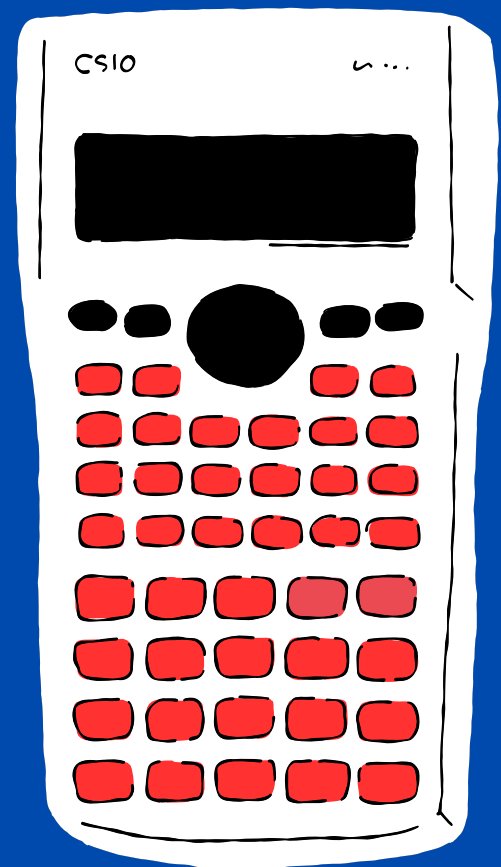
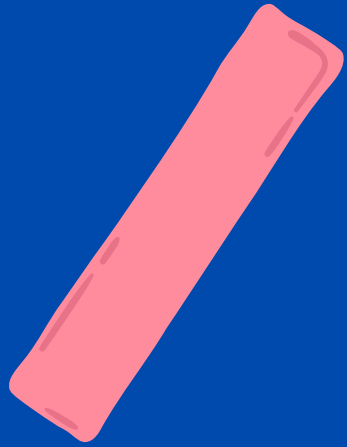
Debe tenerse siempre en cuenta que los operadores actúan de acuerdo con el tipo de dato de los operandos involucrados.

Ejemplo:

```
x = 5 / 2; //el resultado es 2
```

```
x = 5.0 / 2; //el resultado es 2.5
```

C/C++ no cuenta con un operador de potencia, pero se puede utilizar la función `pow()` de la librería `math.h`





OPERADORES INCREMENTALES

Los operadores **incrementales** y **decrementales** en C++ son operadores que se utilizan para **aumentar** o **disminuir** el valor de una variable en una unidad. Estos operadores son muy comunes y se utilizan a menudo en estructuras de control como bucles for, while, etc. Veamos cómo funcionan:

Operadores Incrementales

- Operador ++: Este operador incrementa el valor de la variable en 1.

Operadores Decrementales

- Operador --: Este operador disminuye el valor de la variable en 1.

++

```
int x = 5;
int y;

// Forma prefija
y = ++x; // x se incrementa a 6, y toma el valor de x, por lo tanto y = 6

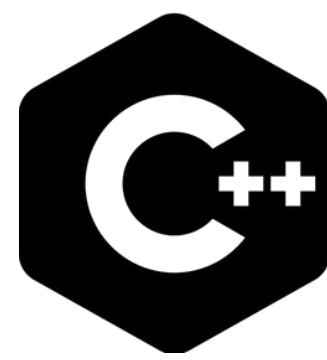
// Forma sufija
y = x++; // y toma el valor de x (que es 6), luego x se incrementa a 7
```

--

```
int x = 5;
int y;

// Forma prefija
y = --x; // x se decrementa a 4, y toma el valor de x, por lo tanto y = 4

// Forma sufija
y = x--; // y toma el valor de x (que es 4), luego x se decrementa a 3
```



OPERADORES CON CADENAS

Para los operadores con cadenas es necesario incluir la Librería:

#include <string.h>

Lo cual nos permitirá hacer operaciones con cadenas.

strcat(Variable1, Variable2)

Concatena Variable1 y Variable2.

strcmp(Variable1, Variable2)

Compara Variable1 y Variable2 retornando Negativo(cero) o positivo para Variable1<Variable2, Variable1==Variable2, Variable1>Variable2.

strcpy(Variable1, Variable2)

Copia el contenido la cadena de Variable2 en Variable.

strlen(Variable)

Retorna la Longitud de la Variable



EXPRESIONES BOOLEANAS

TRUE

FALSE

Una **expresión booleana** es una expresión que evalúa a un valor de tipo booleano, es decir, a **true** (verdadero) o **false** (falso). Estas expresiones son esenciales en estructuras de control como 'if', 'while', 'for', entre otras, **donde se necesita evaluar una condición.**





OPERADORES DE COMPARACIÓN

Una manera de formar **expresiones booleanas** es utilizando los operadores de comparación:

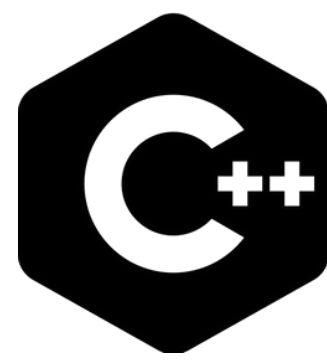
<code>a == b</code>	igualdad
<code>a < b</code>	menor a
<code>a > b</code>	mayor a
<code>a != b</code>	distinto de
<code>a <= b</code>	menor o igual a
<code>a >= b</code>	mayor o igual a

¿Cuál es el Resultado?



```
int a = 5, b = 10;

// Operadores Relacionales
bool resultado1 = (a == b);
bool resultado2 = (a < b);
bool resultado3 = (a >= b);
```



OPERADORES BOOLEANOS

Las expresiones booleanas pueden combinarse mediante los operadores booleanos clásicos:

- **&&** Conjunción (AND)
- **||** Disyunción (OR)
- **!** Negación (NOT)

&& : AND lógico. Devuelve **true** si ambas expresiones son verdaderas.

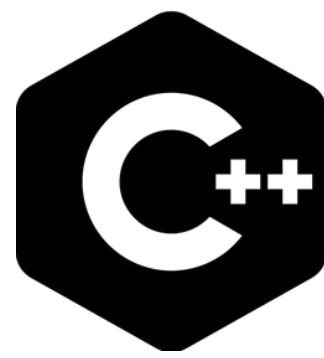
|| : OR lógico. Devuelve **true** si al menos una de las expresiones es verdadera.

! : NOT lógico. **Invierte el valor de la expresión**, es decir, **true** se convierte en **false** y viceversa.

Si a=7 y b=5
¿Qué
obtenemos?



```
// Operadores Lógicos
bool resultado4 = (a < b && b > 0);
bool resultado5 = (a > b || b > 0);
bool resultado6 = !(a < b);
```



EJEMPLOS ARITMETICOS

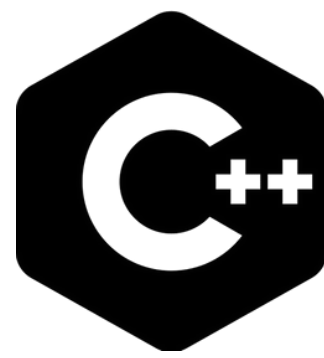
Ejemplo 1

Como Saber cuanto vale mi variable 'y'?

Si **x=2** y **b=6**

$$y = 5x^2 + \frac{2}{3}(b^2 - b)x + b$$

¡Lo Descubramos!



EJEMPLOS ARITMETICOS

Ejemplo 1

Como Saber cuanto vale mi variable 'y'?

Si **x=2** y **b=6**

$$y = 5x^2 + \frac{2}{3}(b^2 - b)x + b$$

Reemplazemos el valor 'x' :

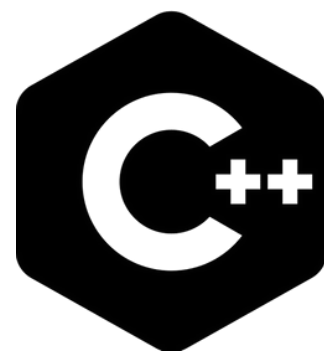
$$y = 5*(2^2) + (2/3)*(b^2 - b)*2 + b$$

Reemplazemos el valor de 'b':

$$y = 5*(2^2) + (2/3)*(6^2 - 6)*2 + 6$$

¿Cuanto vale mi variable 'y' en la calculadora?

¿Lo llevamos a Codigo C++?



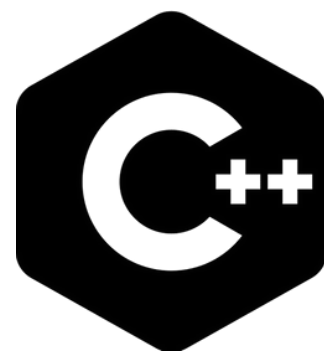
EJEMPLOS ARITMETICOS

Ejemplo 2

Como hacemos ahora con este Problema Aritmetico?

$$y = \frac{1-x}{1+x} + \frac{x^2+7}{1-x^2}$$

¿Cuanto vale 'y'? ¡Pasemos a C++ y lo averiguemos!



EJEMPLOS ARITMETICOS

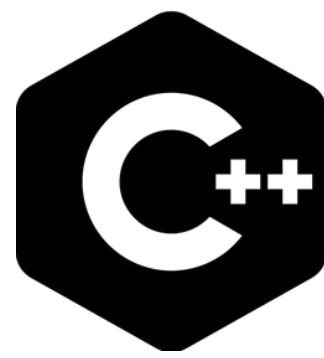
Ejemplo 3

Forma general de la ecuación de 2º grado: $ax^2 + bx + c = 0$

Para la Ecuacion de 2º Grado, tenemos dos valores, x1 y x2

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

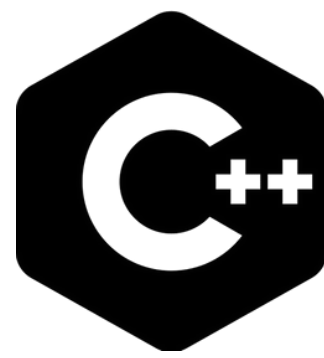
¡Lo Codifiquemos!



EJEMPLOS LOGICOS

Ejemplo 4

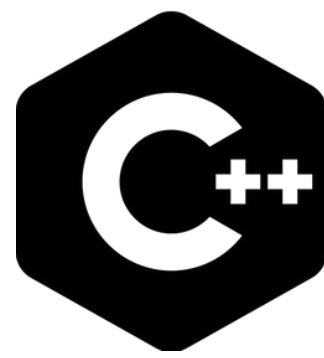
Escribe un programa en C++ que le pida al usuario que ingrese su nombre y su apellido por separado. Luego, el programa debe unir ambas cadenas de texto para formar el nombre completo y mostrarlo en la pantalla, con un espacio entre el nombre y el apellido.



EJEMPLOS LOGICOS

Ejemplo 4

Escribe un programa en C++ que le pida al usuario que ingrese su nombre y su apellido por separado. Luego, el programa debe unir ambas cadenas de texto para formar el nombre completo y mostrarlo en la pantalla, con un espacio entre el nombre y el apellido.

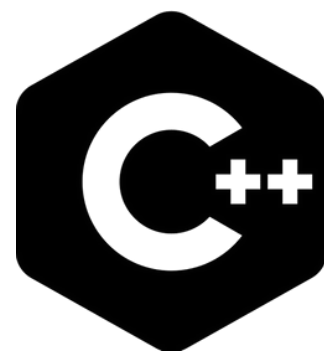


EJEMPLOS LOGICOS

Ejemplo 5

Desarrolla un programa en C++ que trabaje con varias cadenas de caracteres. El programa debe realizar las siguientes operaciones, mostrando el resultado de cada una:

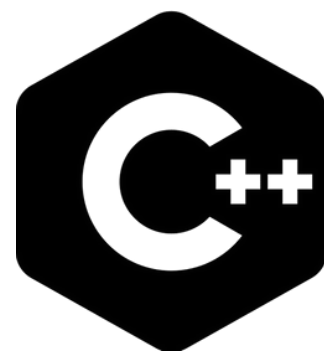
- 1. Declarar al menos dos cadenas de caracteres e inicializarlas con valores.**
- 2. Concatenar una cadena a otra.**
- 3. Copiar el contenido de una cadena en una nueva variable.**
- 4. Comparar dos cadenas para determinar si son iguales.**
- 5. Obtener y mostrar la longitud de una de las cadenas.**



EJEMPLOS LOGICOS

Ejemplo 6

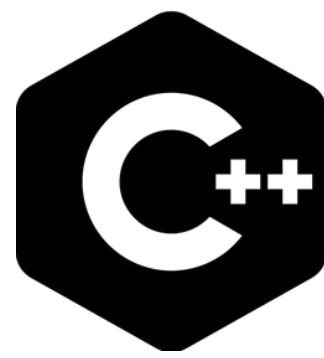
Escribe un programa en C++ que pida al usuario que ingrese una palabra o frase. El programa debe calcular y mostrar la longitud de la cadena (la cantidad de caracteres que tiene) utilizando una función de la librería `<string.h>`.



EJEMPLOS LOGICOS

Ejemplo 7

Escribe un programa en C++ que declare dos variables enteras y les asigne valores. Luego, el programa debe usar operadores relacionales y lógicos para evaluar dos expresiones y mostrar el resultado en la pantalla como 1 (verdadero) o 0 (falso)



EJEMPLOS LOGICOS

Ejemplo 8

Crea un programa en C++ que declare tres variables de tipo char e inicialice sus valores. El programa debe evaluar y mostrar el resultado booleano de dos expresiones lógicas que comparen estas variables. Las expresiones deben utilizar al menos un operador relacional (==, !=, <, >, <=, >=) y al menos un operador lógico (&&, ||, !). El resultado debe mostrarse como 1 para verdadero y 0 para falso

¿PREGUNTAS?

```
render() {  
  return (  
    <React.Fragment>  
      <div className="py-5">  
        <div className="container">  
          <Title name="our" title="product">  
            <div className="row">  
              <ProductConsumer>  
                {(value) => {  
                  console.log(value)  
                }}  
            </ProductConsumer>  
          </div>  
        </div>  
      </div>  
    </React.Fragment>  
  )  
}
```




MUCHAS
GRACIAS!!!

Nos Vemos en la Siguiente Clase