

Clase II:  
Etapas de un programa en C  
Buenas prácticas de programación  
Estructura switch

Informática I  
Centro Regional Universitario Córdoba  
UNDEF

13 de abril de 2021

# Etapas de un programa en C

- 1 Escritura del código fuente: se pueden utilizar diferentes editores de texto: gedit, VIM, Zinjal, Atom, Emacs, etc.

► [Ver en github](#)

# Etapas de un programa en C

- 1 Escritura del código fuente: se pueden utilizar diferentes editores de texto: gedit, VIM, Zinjal, Atom, Emacs, etc.  
[▶ Ver en github](#)
- 2 Pre-procesamiento: procesa directivas como `#include`, `#define` e `#if`, remueve los comentarios, etc. En general, luego del preprocesamiento, el archivo resultante contiene una gran cantidad de líneas de código.

[▶ Ver en github](#)

# Etapas de un programa en C

- 1 Escritura del código fuente: se pueden utilizar diferentes editores de texto: gedit, VIM, Zinjal, Atom, Emacs, etc.  
[▶ Ver en github](#)
- 2 Pre-procesamiento: procesa directivas como `#include`, `#define` e `#if`, remueve los comentarios, etc. En general, luego del preprocesamiento, el archivo resultante contiene una gran cantidad de líneas de código.  
[▶ Ver en github](#)
- 3 Compilado: el compilador de C traduce el código del apartado anterior a assembler.  
[▶ Ver en github](#)

# Etapas de un programa en C

- 1 Escritura del código fuente: se pueden utilizar diferentes editores de texto: gedit, VIM, Zinjal, Atom, Emacs, etc.  
[▶ Ver en github](#)
- 2 Pre-procesamiento: procesa directivas como `#include`, `#define` e `#if`, remueve los comentarios, etc. En general, luego del preprocesamiento, el archivo resultante contiene una gran cantidad de líneas de código.  
[▶ Ver en github](#)
- 3 Compilado: el compilador de C traduce el código del apartado anterior a assembler.  
[▶ Ver en github](#)
- 4 Ensamblado: el ensamblado transforma el programa escrito en lenguaje ensamblador a código objeto.  
[▶ Ver en github](#)

# Etapas de un programa en C

- 1 Escritura del código fuente: se pueden utilizar diferentes editores de texto: gedit, VIM, Zinjal, Atom, Emacs, etc.  
▶ [Ver en github](#)
- 2 Pre-procesamiento: procesa directivas como `#include`, `#define` e `#if`, remueve los comentarios, etc. En general, luego del preprocesamiento, el archivo resultante contiene una gran cantidad de líneas de código.  
▶ [Ver en github](#)
- 3 Compilado: el compilador de C traduce el código del apartado anterior a assembler.  
▶ [Ver en github](#)
- 4 Ensamblado: el ensamblado transforma el programa escrito en lenguaje ensamblador a código objeto.  
▶ [Ver en github](#)
- 5 Enlazado: reúne uno o más módulos en código objeto con el código existente en las bibliotecas.  
▶ [Ver en github](#)

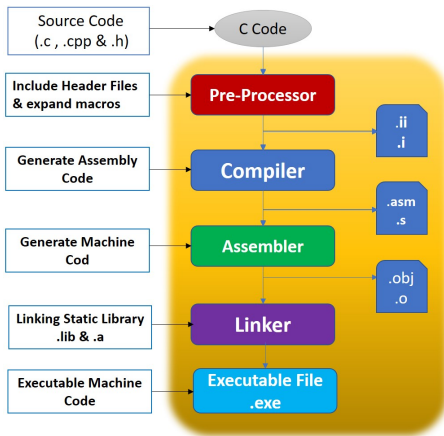


Figura: Etapas de un programa en C.

Estos ejemplos fueron generados agregando el flag `save-temps`.<sup>a1</sup> al compilador gcc.

# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 líneas de código o menos
  - Aplicación promedio para smartphone



# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 lineas de código o menos
  - Aplicación promedio para smartphone
  - Photoshop v1.0 (1990)

# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 líneas de código o menos
  - Aplicación promedio para smartphone
  - Photoshop v1.0 (1990)
  - Quake 3 (1999)

# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 líneas de código o menos
  - Aplicación promedio para smartphone
  - Photoshop v1.0 (1990)
  - Quake 3 (1999)
  - Transbordador espacial STS (1981)

# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 líneas de código o menos
  - Aplicación promedio para smartphone
  - Photoshop v1.0 (1990)
  - Quake 3 (1999)
  - Transbordador espacial STS (1981)
- Proyectos con aproximadamente menos de 10 millones de líneas de código

# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 líneas de código o menos
  - Aplicación promedio para smartphone
  - Photoshop v1.0 (1990)
  - Quake 3 (1999)
  - Transbordador espacial STS (1981)
- Proyectos con aproximadamente menos de 10 millones de líneas de código
  - Age of empires - Versión online

# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 líneas de código o menos
  - Aplicación promedio para smartphone
  - Photoshop v1.0 (1990)
  - Quake 3 (1999)
  - Transbordador espacial STS (1981)
- Proyectos con aproximadamente menos de 10 millones de líneas de código
  - Age of empires - Versión online
  - Linux kernel 2.2.0

# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 líneas de código o menos
  - Aplicación promedio para smartphone
  - Photoshop v1.0 (1990)
  - Quake 3 (1999)
  - Transbordador espacial STS (1981)
- Proyectos con aproximadamente menos de 10 millones de líneas de código
  - Age of empires - Versión online
  - Linux kernel 2.2.0
  - Windows 3.1 (1992)

# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 líneas de código o menos
  - Aplicación promedio para smartphone
  - Photoshop v1.0 (1990)
  - Quake 3 (1999)
  - Transbordador espacial STS (1981)
- Proyectos con aproximadamente menos de 10 millones de líneas de código
  - Age of empires - Versión online
  - Linux kernel 2.2.0
  - Windows 3.1 (1992)
- Proyectos con aproximadamente más de 50 millones



# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 líneas de código o menos
  - Aplicación promedio para smartphone
  - Photoshop v1.0 (1990)
  - Quake 3 (1999)
  - Transbordador espacial STS (1981)
- Proyectos con aproximadamente menos de 10 millones de líneas de código
  - Age of empires - Versión online
  - Linux kernel 2.2.0
  - Windows 3.1 (1992)
- Proyectos con aproximadamente más de 50 millones
  - Age of empires - Versión online

# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 líneas de código o menos
  - Aplicación promedio para smartphone
  - Photoshop v1.0 (1990)
  - Quake 3 (1999)
  - Transbordador espacial STS (1981)
- Proyectos con aproximadamente menos de 10 millones de líneas de código
  - Age of empires - Versión online
  - Linux kernel 2.2.0
  - Windows 3.1 (1992)
- Proyectos con aproximadamente más de 50 millones
  - Age of empires - Versión online
  - Linux kernel 2.2.0

# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 líneas de código o menos
  - Aplicación promedio para smartphone
  - Photoshop v1.0 (1990)
  - Quake 3 (1999)
  - Transbordador espacial STS (1981)
- Proyectos con aproximadamente menos de 10 millones de líneas de código
  - Age of empires - Versión online
  - Linux kernel 2.2.0
  - Windows 3.1 (1992)
- Proyectos con aproximadamente más de 50 millones
  - Age of empires - Versión online
  - Linux kernel 2.2.0
  - Windows 3.1 (1992)

# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 líneas de código o menos
  - Aplicación promedio para smartphone
  - Photoshop v1.0 (1990)
  - Quake 3 (1999)
  - Transbordador espacial STS (1981)
- Proyectos con aproximadamente menos de 10 millones de líneas de código
  - Age of empires - Versión online
  - Linux kernel 2.2.0
  - Windows 3.1 (1992)
- Proyectos con aproximadamente más de 50 millones
  - Age of empires - Versión online
  - Linux kernel 2.2.0
  - Windows 3.1 (1992)
- Proyectos con aproximadamente más de 2 mil millones de líneas de código

# Buenas practicas de programación

- Proyectos con aproximadamente 300.000 líneas de código o menos
  - Aplicación promedio para smartphone
  - Photoshop v1.0 (1990)
  - Quake 3 (1999)
  - Transbordador espacial STS (1981)
- Proyectos con aproximadamente menos de 10 millones de líneas de código
  - Age of empires - Versión online
  - Linux kernel 2.2.0
  - Windows 3.1 (1992)
- Proyectos con aproximadamente más de 50 millones
  - Age of empires - Versión online
  - Linux kernel 2.2.0
  - Windows 3.1 (1992)
- Proyectos con aproximadamente más de 2 mil millones de líneas de código
  - Google Internet Services

# Buenas practicas de programación: reglas de estilo

- Variables

# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos

# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos
  - Si el nombre contiene dos o más palabras, se las debe separar por \_



# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos
  - Si el nombre contiene dos o más palabras, se las debe separar por \_
  - Deben inicializarse a un valor conocido al momento de la declaración

# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos
  - Si el nombre contiene dos o más palabras, se las debe separar por \_
  - Deben inicializarse a un valor conocido al momento de la declaración
- Constantes definidas con # define

# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos
  - Si el nombre contiene dos o más palabras, se las debe separar por \_
  - Deben inicializarse a un valor conocido al momento de la declaración
- Constantes definidas con # define
  - Su identificador debe escribirse con mayúsculas

# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos
  - Si el nombre contiene dos o más palabras, se las debe separar por \_
  - Deben inicializarse a un valor conocido al momento de la declaración
- Constantes definidas con # define
  - Su identificador debe escribirse con mayúsculas
- Indentación del código

# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos
  - Si el nombre contiene dos o más palabras, se las debe separar por \_
  - Deben inicializarse a un valor conocido al momento de la declaración
- Constantes definidas con # define
  - Su identificador debe escribirse con mayúsculas
- Indentación del código
  - Debe realizarse con tabulaciones, nunca con espacios

# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos
  - Si el nombre contiene dos o más palabras, se las debe separar por \_
  - Deben inicializarse a un valor conocido al momento de la declaración
- Constantes definidas con # define
  - Su identificador debe escribirse con mayúsculas
- Indentación del código
  - Debe realizarse con tabulaciones, nunca con espacios
  - Deben colocarse llaves para demarcar gráficamente el código que ejecutará cada estructura

# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos
  - Si el nombre contiene dos o más palabras, se las debe separar por \_
  - Deben inicializarse a un valor conocido al momento de la declaración
- Constantes definidas con # define
  - Su identificador debe escribirse con mayúsculas
- Indentación del código
  - Debe realizarse con tabulaciones, nunca con espacios
  - Deben colocarse llaves para demarcar gráficamente el código que ejecutará cada estructura
  - Las llaves deben escribirse siempre

# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos
  - Si el nombre contiene dos o más palabras, se las debe separar por \_
  - Deben inicializarse a un valor conocido al momento de la declaración
- Constantes definidas con # define
  - Su identificador debe escribirse con mayúsculas
- Indentación del código
  - Debe realizarse con tabulaciones, nunca con espacios
  - Deben colocarse llaves para demarcar gráficamente el código que ejecutará cada estructura
  - Las llaves deben escribirse siempre
  - Las líneas de código no deben superar la longitud de 80 caracteres



# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos
  - Si el nombre contiene dos o más palabras, se las debe separar por \_
  - Deben inicializarse a un valor conocido al momento de la declaración
- Constantes definidas con # define
  - Su identificador debe escribirse con mayúsculas
- Indentación del código
  - Debe realizarse con tabulaciones, nunca con espacios
  - Deben colocarse llaves para demarcar gráficamente el código que ejecutará cada estructura
  - Las llaves deben escribirse siempre
  - Las líneas de código no deben superar la longitud de 80 caracteres
- Instrucciones que se deben evitar

# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos
  - Si el nombre contiene dos o más palabras, se las debe separar por \_
  - Deben inicializarse a un valor conocido al momento de la declaración
- Constantes definidas con # define
  - Su identificador debe escribirse con mayúsculas
- Indentación del código
  - Debe realizarse con tabulaciones, nunca con espacios
  - Deben colocarse llaves para demarcar gráficamente el código que ejecutará cada estructura
  - Las llaves deben escribirse siempre
  - Las líneas de código no deben superar la longitud de 80 caracteres
- Instrucciones que se deben evitar
  - goto

# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos
  - Si el nombre contiene dos o más palabras, se las debe separar por \_
  - Deben inicializarse a un valor conocido al momento de la declaración
- Constantes definidas con # define
  - Su identificador debe escribirse con mayúsculas
- Indentación del código
  - Debe realizarse con tabulaciones, nunca con espacios
  - Deben colocarse llaves para demarcar gráficamente el código que ejecutará cada estructura
  - Las llaves deben escribirse siempre
  - Las líneas de código no deben superar la longitud de 80 caracteres
- Instrucciones que se deben evitar
  - goto
  - Break

# Buenas practicas de programación: reglas de estilo

- Variables
  - Los nombres deben ser cortos, descriptivos y concretos
  - Si el nombre contiene dos o más palabras, se las debe separar por \_
  - Deben inicializarse a un valor conocido al momento de la declaración
- Constantes definidas con # define
  - Su identificador debe escribirse con mayúsculas
- Indentación del código
  - Debe realizarse con tabulaciones, nunca con espacios
  - Deben colocarse llaves para demarcar gráficamente el código que ejecutará cada estructura
  - Las llaves deben escribirse siempre
  - Las líneas de código no deben superar la longitud de 80 caracteres
- Instrucciones que se deben evitar
  - goto
  - Break
  - Continue

# Estructura Switch

## Aplicación

Permite que un programa en C tome diferentes caminos en función del valor que tome una determinada instrucción.

## Pseudocódigo

Según sea (variable de control)

Caso 1:

Instrucciones caso 1

Instrucciones caso 1

frenar

Caso 2:

Instrucciones caso 2

Instrucciones caso 2

frenar

Caso por descarte

Instrucciones caso por descarte

frenar

# Estructura Switch

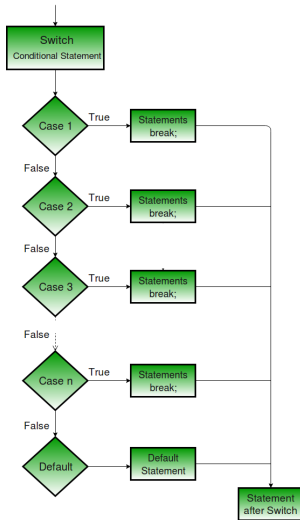


Figura: Diagrama de flujo para la estructura switch

# Estructura Switch

```
1  #include <stdio.h>
2  int main()
3  {   int opcion=0;
4      printf(" Ingrese una numero entre 0-2\n" );
5      scanf(" %d",&opcion );
6      switch(opcion){
7          case 0:
8              printf(" Usted ingreso la opcion 0 \n" );
9              break;
10         case 1:
11             printf(" Usted ingreso la opcion 1 \n" );
12             break;
13         case 2:
14             printf(" Usted ingreso la opcion 2 \n" );
15             break;
16         default :
17             printf(" Usted ingreso una opcion invalida\n"
18             break;}
19
20     return (0);
```

# Estructura Switch - ejemplos I

- 1 Diseñar y codificar un programa que permita tomar por teclado un número entero comprendido entre 0 y 2 inclusive. Luego el programa imprimir por pantalla la opción seleccionada en modo texto. Si el usuario ingresa una opción invalida, debe informarse dicha excepción por pantalla.

► [Ver en github](#)

- 2 Diseñar y codificar un programa que permita tomar por teclado un número entero. Este número entero representa un mes del año, siendo 1 para enero y 12 para diciembre. El programa debe imprimir por pantalla la cantidad de días del mes seleccionado. <sup>1</sup>

► [Ver en github](#)



- 3 Diseñar y codificar un programa que permita conocer el estado de un alumno en función de la nota de su parcial. Si la nota ingresada es menor a cuatro, se debe imprimir reprobado. Entre cuatro y diez, aprobado. Cualquier otra opción, imprimir mensaje indicando que la nota es incorrecta. Se debe usar una estructura selectiva switch.

► [Ver en github](#)

---

<sup>1</sup>Se supone un año no bisiesto.