ALGUNOS CONSEJOS ADICIONALES SOBRE C Y C++

- No usar `gets`. Es muy peligroso. Por ejemplo: no es posible saber por adelantado cuantos caracteres hay que leer; `gets` puede continuar almacenando caracteres después del final del "buffer". Es preferible usar `fgets` o `scanf`.

```
// ejemplo de uso de fgets en lugar de gets que es peligroso

#include <stdio.h>
#include <string.h>
int main()
{
  char nombre[21];
  int caso, longi;
  char c;

  caso = 3;

  switch(caso){
  case 1:
    printf("\n\nPor favor: digite su nombre: ");
    gets(nombre);
    printf("\nGracias %s. Hasta pronto.\n\n", nombre);
    break;
  case 2:
    // stdin es el flujo estandar de entrada
    // (standard input)
    printf("\n\nPor favor: digite su nombre: ");
    fgets(nombre, 20, stdin);
    printf("\nGracias %s. Hasta pronto.\n\n", nombre);
    break;
  case 3:
    // fgets, en la lectura por teclado de una cadena,
    // tambien lee el simbolo de fin de linea y lo coloca al final.
    // Generalmente, es conveniente quitar ese caracter.
    printf("\n\nPor favor: digite su nombre: ");
    fgets(nombre, 20, stdin);
    longi = strlen(nombre);
    if( nombre[longi-1] == '\n' ) nombre[longi-1] = '\0';
    printf("\nGracias %s. Hasta pronto.\n\n", nombre);
    break;
  case 4:
    // OJO: cuando se lee una cadena con  scanf
```

```
    // y la cadena contiene espacios en blanco,
    // alli se corta la cadena y desprecia lo que sigue.
    printf("\n\nPor favor: digite su nombre: ");
    scanf("%s", nombre);
    printf("\nGracias %s. Hasta pronto.\n\n", nombre);
    break;
  }

  return 0;
}
```

- Puede haber problemas al mezclar en el mismo programa órdenes de entrada de `stdio.h` con órdenes asociadas a la lectura en el flujo (*stream*). Ejemplos con algún riesgo: `scanf` y `cin`; `scanf` y `gets`;

  //Un consejo general que evita algunos inconvenientes es: //no mezclar en un mismo programa ordenes de entrada de datos de `stdio.h` //con órdenes de entrada `iostream`.

- Algunas veces es necesario hacer pausas durante la ejecución de un programa o al final de éste. Algunos compiladores tienen funciones que son muy útiles para lograr este objetivo. Por ejemplo la función `getche` de los compiladores de Borland. Su uso no es conveniente si se desea generalidad y portabilidad. En algunas plataformas funciona bien la orden

  ```
  system("pause");
  ```

  usando el archivo de cabecera `stdlib.h`.

  A continuación hay varios ejemplos de funciones de pausa.

```
void pausa1()
{
  char c;

  printf("Oprima ENTER para continuar. ");
  scanf("%c", &c);
}
//-----------------------------------------------------------
void pausa2()
{
  char c;

  printf("Oprima ENTER para continuar. ");
  c = getc(stdin);
}
//-----------------------------------------------------------
void pausa3()
{
```

```
  char c;

  printf("Oprima ENTER para continuar. ");
  c = getchar();
}
//-------------------------------------------------------
void pausa4()
{
  char c[3];

  printf("Oprima ENTER para continuar. ");
  fgets(c, 2, stdin);
}
//-------------------------------------------------------
void pausa12()
{
  char c;

  cout<<"Oprima ENTER para continuar. ";
  c = getc(stdin);
}
//-------------------------------------------------------
void pausa13()
{
  char c;

  cout<<"Oprima ENTER para continuar. ";
  c = getchar();
}
//-------------------------------------------------------
void pausa15()
{
  char c[3];

  cout<<"Oprima ENTER para continuar. ";
  fgets(c, 2, stdin);
}
```