I OBJETIVOS

- Medir y comparar la ejecución de procesos que requieren uso intensivo del CPU en arquitecturas Multi-Core.
- Utilizar la API Windows para la gestión de hilos, y de esta forma optimizar una aplicación que requiere uso intensivo del CPU para procesadores Multi-Core.

II BIBLIOGRAFÍA

- Manuales de programación con la API de Windows.

III RECURSOS

- Una estación de trabajo con Visual Studio 2005/2008 o 2010.
- Computadoras con procesadores Multi-Core.

IV ACTIVIDADES

1 Procesamiento de una imagen BMP

1.1 El formato BMP

El formato BMP es uno de los más simples. Fue desarrollado por Microsoft e IBM en forma conjunta, lo que explica su uso particularmente amplio en plataformas Windows y OS/2. Un archivo BMP es un archivo de mapa de bits, es decir, un archivo de imagen de gráficos, con píxeles almacenados en forma de tabla de puntos que administra los colores como colores reales o usando una paleta indexada. El formato BMP ha sido estudiado de manera tal que permite obtener un mapa de bits independiente del dispositivo de visualización periférico (DIB, Mapa de bits independiente del dispositivo).

La estructura de un mapa de bits es la siguiente:

- Encabezado del archivo
- Encabezado de información del mapa de bits (también llamado encabezado de información).

- Paleta (opcional)
- Cuerpo de la imagen

Encabezado del archivo

El encabezado del archivo proporciona información acerca del tipo de archivo (mapa de bits) y su tamaño, así como también indica dónde comienza realmente la información de la imagen.

El encabezado comprende cuatro campos:

- La firma (en 2 bytes), que indica que se trata de un archivo BMP con dos caracteres
  - BM, 424D en hexadecimal, que indica que se trata de un mapa de bits de Windows
  - BA que indica que se trata de un mapa de bits OS/2
  - CI que indica que se trata de un icono de color de OS/2
  - CP indica que es un puntero de color de OS/2
  - IC indica que es un icono de OS/2
  - PT indica que es un puntero de OS/2
- El tamaño total del archivo en bytes (codificado en 4 bytes)
- Un campo reservado (en 4 bytes)
- El desajuste de la imagen (en 4 bytes), es decir, la ubicación del comienzo de la información de la imagen en relación con el comienzo del archivo

Encabezado de información del mapa de bits

El encabezado de información del mapa de bits proporciona información acerca de la imagen, en especial las dimensiones y los colores.

La información del mapa de bits comprende cuatro campos:

- El tamaño del encabezado de información del mapa de bits en bytes (codificado en 4 bytes). Los siguientes valores hexadecimales son posibles según el tipo de formato BMP:
  - 28 para Windows 3.1x, 95, NT
  - 0C para OS/2 1.x

    - F0 para OS/2 2.x
- El ancho de la imagen (en 4 bytes), es decir, el número de píxeles contados de forma horizontal
- La altura de la imagen (en 4 bytes), es decir, el número de píxeles contados de forma vertical
- El número de planos (en 2 bytes). Este valor es siempre 1
- La profundidad del modelo de color (en 2 bytes), es decir, el número de bits usados para codificar el color. Este valor puede ser equivalente a 1, 4, 8, 16, 24 ó 32
- El método de compresión (en 4 bytes). Este valor es 0 cuando la imagen no está comprimida o 1, 2 ó 3 según el tipo de compresión usado:
    - 1 para la codificación RLE de 8 bits por píxel
    - 2 para la codificación RLE de 4 bits por píxel
    - 3 para la codificación de campo de bits, lo que significa que el color fue codificado por una máscara triple representada por la paleta
- El tamaño total de la imagen en bytes (en 4 bytes).
- La resolución horizontal (en 4 bytes), es decir, el número de píxeles por metro contado de forma horizontal
- La resolución vertical (en 4 bytes), es decir, el número de píxeles por metro contado de forma vertical
- El número de colores de la paleta (en 4 bytes)
- El número de colores importantes de la paleta (en 4 bytes). Este campo puede equivaler a 0 cuando todos los colores son importantes.

Paleta de imágenes

La paleta es opcional. Cuando se define la paleta, ésta contiene 4 bytes de forma sucesiva para cada una de las entradas, que representan:

- El componente azul (en un byte)
- El componente verde (en un byte)
- El componente rojo (en un byte)
- Un campo reservado (en un byte)

Codificación de imágenes

La codificación de imágenes se realiza escribiendo en forma sucesiva los bits que corresponden a cada píxel, línea por línea, comenzando por el píxel del extremo inferior izquierdo.

- Las imágenes de 2 colores usan 1 bit por píxel, lo que significa que un byte permite codificar 8 píxeles
- Las imágenes de 16 colores usan 4 bits por píxel, lo que significa que un byte permite codificar 2 píxeles
- Las imágenes de 256 colores usan 8 bits por píxel, lo que significa que se necesita un byte para codificar cada píxel
- Las imágenes de colores reales usan 24 bits por píxel, lo que significa que se necesitan 3 bytes para codificar cada píxel, respetando la alternancia del orden de los colores para el azul, el verde y el rojo.

Cada línea de la imagen debe comprender un número total de bytes que sea múltiplo de 4; si este esquema no se cumple, la línea se debe completar con todos los 0 necesarios para respetar el criterio.

2 Detección de bordes en una imagen BMP

El programa *Detección de bordes* que se muestra en el Ejemplo 1 lee un archivo .BMP y a partir de ahí genera otro archivo .BMP detectando los bordes de las imágenes. Para la detección es necesario recorrer cada uno de los pixeles de la imagen original y comparar su valor con el valor de su sus 8 vecinos. Los vecinos de un pixel *P* son todos aquellos que lo rodean como podemos observar en la Figura 1.

V1	V2	V3
V4	P	V5
V6	V7	V8

Figura 1.- Un pixel y sus 8 vecinos

Los valores que se comparan de cada pixel son calculados a partir de equivalente en escala de grises, si el valor en escala de grises del pixel P es mayor o igual a la constante `DIF`, entonces tenemos que escribir un pixel de color negro, es decir, los valores de verde, rojo y azul en 0. De lo contrario tenemos que escribir un valor en blanco, los valores de verde, rojo y azul en 255.

El tiempo de ejecución del programa es el tiempo que le toma cargar una imagen a memoria, recorrer los pixeles de la imagen procesando y generando la imagen destino, más escribir la imagen destino en disco.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>

#define DIF 16

// NOMBRE DEL ARCHIVO A PROCESAR
char filename[]="C:\\Users\\José Luis\\Pictures\\pruebas\\imagen.bmp";

#pragma pack(2)  // Empaquetado de 2 bytes
typedef struct {
   unsigned char magic1;   // 'B'
   unsigned char magic2;   // 'M'
   unsigned int size;      // Tamaño
   unsigned short int reserved1, reserved2;
   unsigned int pixelOffset; // offset a la imagen
} HEADER;

#pragma pack() // Empaquetamiento por default
typedef struct {
   unsigned int size;    // Tamaño de este encabezado INFOHEADER
   int cols, rows;       // Renglones y columnas de la imagen
   unsigned short int planes;
   unsigned short int bitsPerPixel; // Bits por pixel
   unsigned int compression;
   unsigned int cmpSize;
   int xScale, yScale;
   unsigned int numColors;
   unsigned int importantColors;
} INFOHEADER;

typedef struct {
      unsigned char red;
      unsigned char green;
      unsigned char blue;
} PIXEL;


typedef struct {
      HEADER header;
      INFOHEADER infoheader;
      PIXEL *pixel;
} IMAGE;

IMAGE imagenfte,imagendst;

int loadBMP(char *filename, IMAGE *image)
{

   FILE *fin;
```

```
   int i=0;
   int totpixs=0;

   fin = fopen(filename, "rb+");

   // Si el archivo no existe
   if (fin == NULL)
            return(-1);

   // Leer encabezado
   fread(&image->header, sizeof(HEADER), 1, fin);

   // Probar si es un archivo BMP
   if (!((image->header.magic1 == 'B') && (image->header.magic2 == 'M')))
            return(-1);

   fread(&image->infoheader, sizeof(INFOHEADER), 1, fin);

   // Probar si es un archivo BMP 24 bits no compactado
   if (!((image->infoheader.bitsPerPixel == 24) && !image-
>infoheader.compression))
         return(-1);

   image->pixel=(PIXEL *)malloc(sizeof(PIXEL)*image->infoheader.cols*image-
>infoheader.rows);

   totpixs=image->infoheader.rows*image->infoheader.cols;

   while(i<totpixs)
   {
            fread(image->pixel+i, sizeof(PIXEL),512, fin);
            i+=512;
   }

   fclose(fin);
}


int saveBMP(char *filename, IMAGE *image)
{
   FILE *fout;
   int i,totpixs;

   fout=fopen(filename,"wb");
   if (fout == NULL)
         return(-1);    // Error

   // Escribe encabezado
   fwrite(&image->header, sizeof(HEADER), 1, fout);


   // Escribe información del encabezado
   fwrite(&image->infoheader, sizeof(INFOHEADER), 1, fout);

   i=0;
```

```
   totpixs=image->infoheader.rows*image->infoheader.cols;
   while(i<totpixs)
   {
            fwrite(image->pixel+i,sizeof(PIXEL),512,fout);
            i+=512;
   }


   fclose(fout);
}


unsigned char blackandwhite(PIXEL p)
{
      return((unsigned char) (0.3*((float)p.red)+
0.59*((float)p.green)+0.11*((float)p.blue)));
}

void processBMP(IMAGE *imagefte, IMAGE *imagedst)
{
      int i,j;
      int count=0;
      PIXEL *pfte,*pdst;
      PIXEL *v0,*v1,*v2,*v3,*v4,*v5,*v6,*v7;
      int imageRows,imageCols;

      memcpy(imagedst,imagefte,sizeof(IMAGE)-sizeof(PIXEL *));

      imageRows = imagefte->infoheader.rows;
      imageCols = imagefte->infoheader.cols;

      imagedst->pixel=(PIXEL *)malloc(sizeof(PIXEL)*imageRows*imageCols);

      for(i=1;i<imageRows-1;i++)
            for(j=1;j<imageCols-1;j++)
            {
                  pfte=imagefte->pixel+imageCols*i+j;
                  v0=pfte-imageCols-1;
                  v1=pfte-imageCols;
                  v2=pfte-imageCols+1;
                  v3=pfte-1;
                  v4=pfte+1;
                  v5=pfte+imageCols-1;
                  v6=pfte+imageCols;
                  v7=pfte+imageCols+1;

                  pdst=imagedst->pixel+imageCols*i+j;

                  if(abs(blackandwhite(*pfte)-blackandwhite(*v0))>DIF ||
                     abs(blackandwhite(*pfte)-blackandwhite(*v1))>DIF ||
                     abs(blackandwhite(*pfte)-blackandwhite(*v2))>DIF ||
                     abs(blackandwhite(*pfte)-blackandwhite(*v3))>DIF ||
                     abs(blackandwhite(*pfte)-blackandwhite(*v4))>DIF ||
                     abs(blackandwhite(*pfte)-blackandwhite(*v5))>DIF ||
                     abs(blackandwhite(*pfte)-blackandwhite(*v6))>DIF ||
                     abs(blackandwhite(*pfte)-blackandwhite(*v7))>DIF)
```

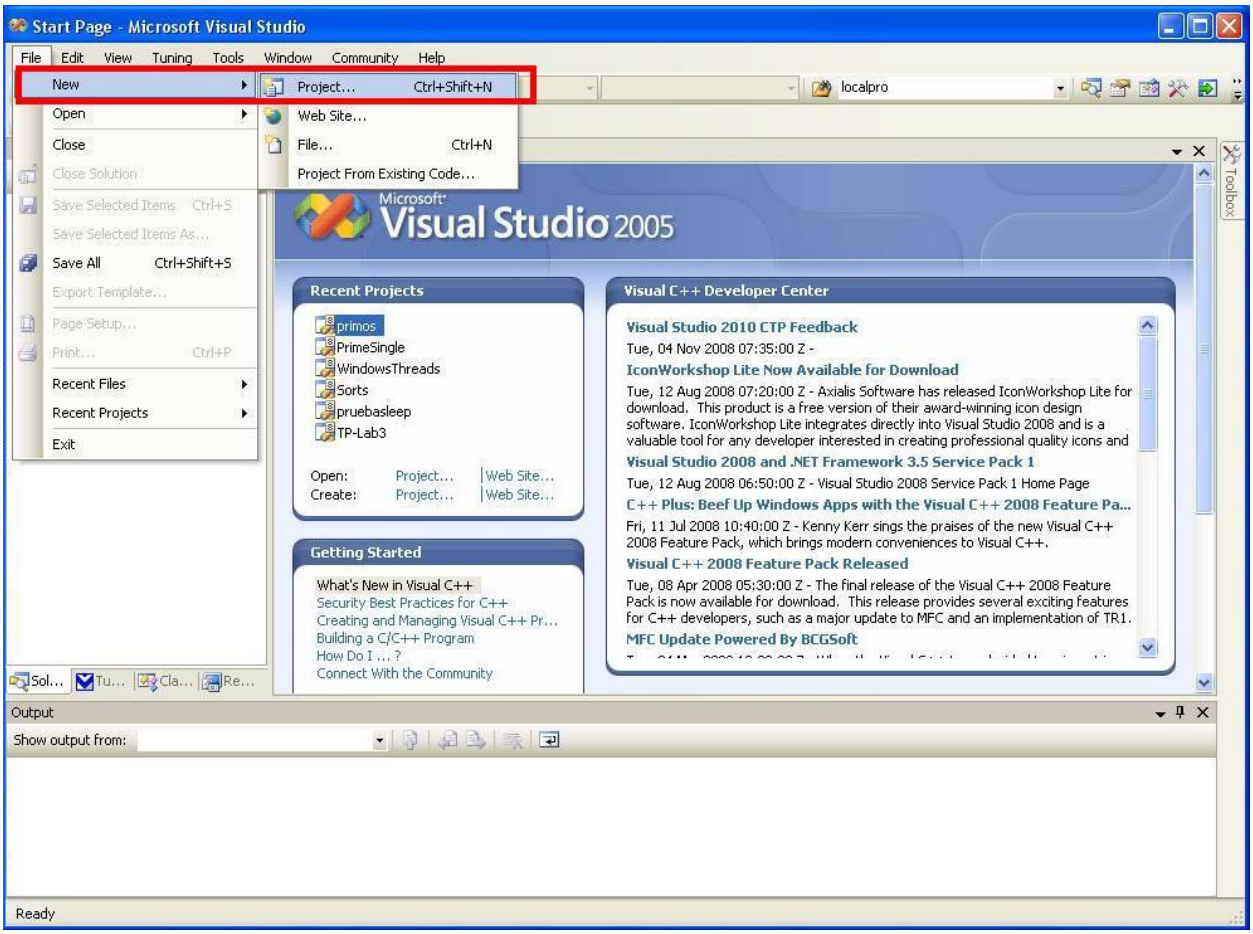
```
                    {
                          pdst->red=0;
                          pdst->green=0;
                          pdst->blue=0;
                    }
                    else
                    {
                          pdst->red=255;
                          pdst->green=255;
                          pdst->blue=255;
                    }
             }
}

int main()
{
      int res;
      clock_t t_inicial,t_final;
      char namedest[80];

      t_inicial=clock();

      strcpy(namedest,strtok(filename,"."));

      strcat(filename,".bmp");
      strcat(namedest,"_P.bmp");
      printf("Archivo fuente %s\n",filename);
      printf("Archivo destino %s\n",namedest);

      res=loadBMP(filename,&imagenfte);
      if(res==-1)
      {
            fprintf(stderr,"Error al abrir imagen\n");
            exit(1);
      }

      printf("Procesando imagen de: Renglones = %d, Columnas =
%d\n",imagenfte.infoheader.rows,imagenfte.infoheader.cols);

      processBMP(&imagenfte,&imagendst);

      res=saveBMP(namedest,&imagendst);
      if(res==-1)
      {
            fprintf(stderr,"Error al escribir imagen\n");
            exit(1);
      }
      t_final=clock();

      printf("Tiempo %3.6f segundos\n",((float) t_final- (float)t_inicial)/
CLOCKS_PER_SEC);
}
```

Ejemplo 1.- Encontrando los bordes dentro de una imagen BMP

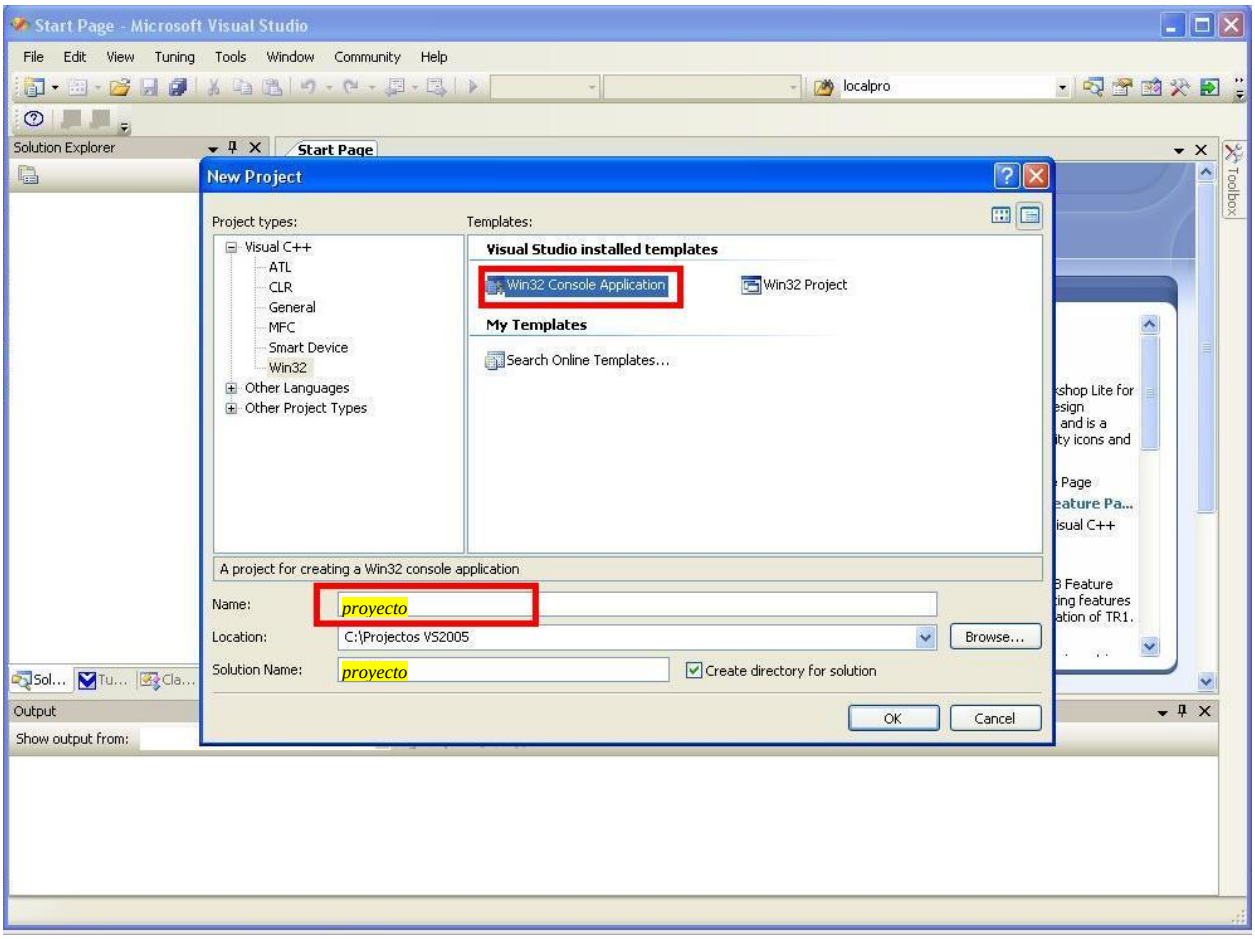
2.1 Editar y compilar un programa en Visual Studio

A continuación se muestra una breve guía para editar y compilar un programa desde Visual Studio 2008. En este caso el ejemplo se basa en el de los números primos por lo que es el nombre que se usa para los proyectos y archivos, pero este puede usarse con cualquier programa.

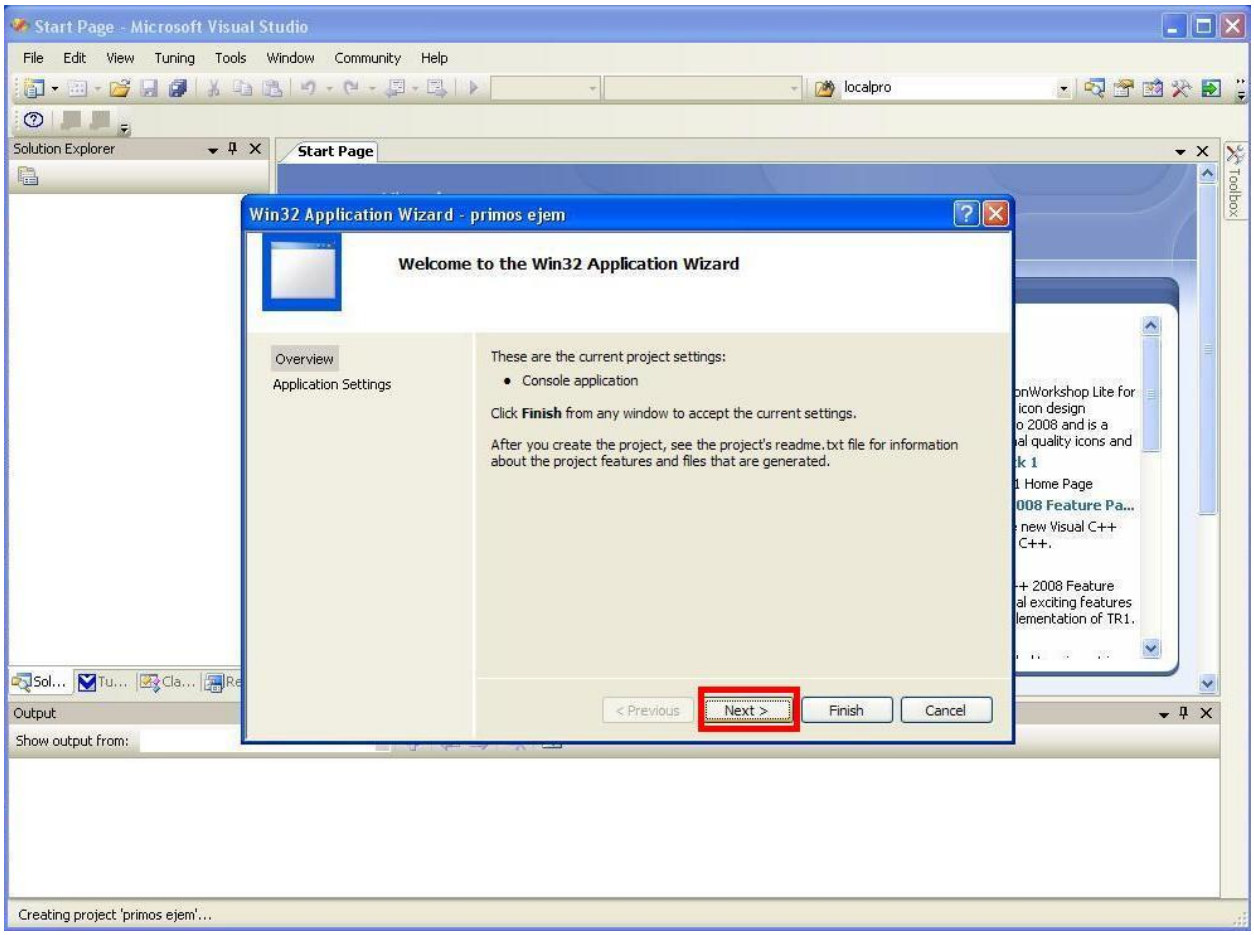
1.- Entrar a Visual Studio 2005/2008/2010. En el menú *File* seleccionar *New* y después *Project*.



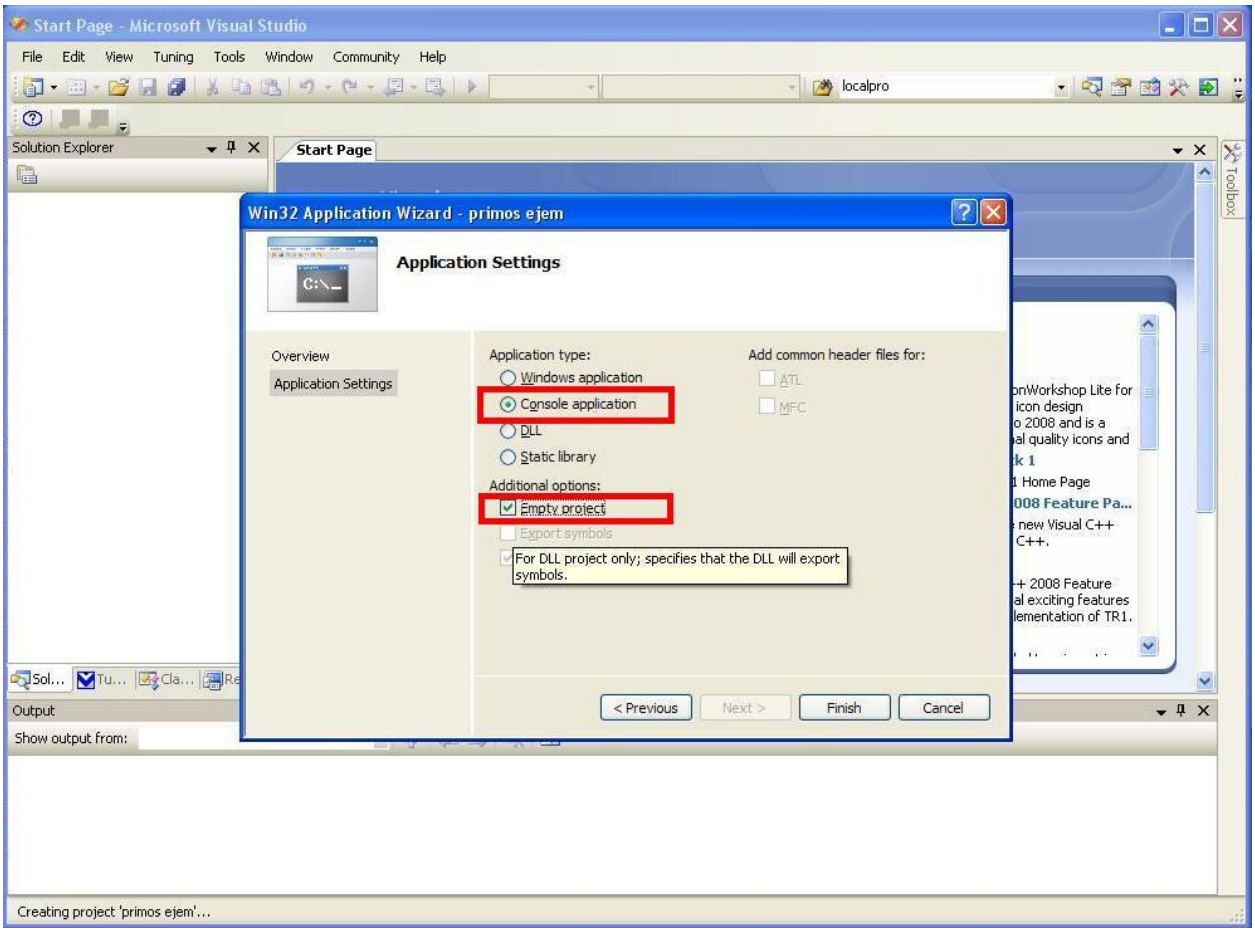
2.- Seleccionar *Win32 Console Application* y en la casilla *Name* introducir el nombre que tendrá el proyecto



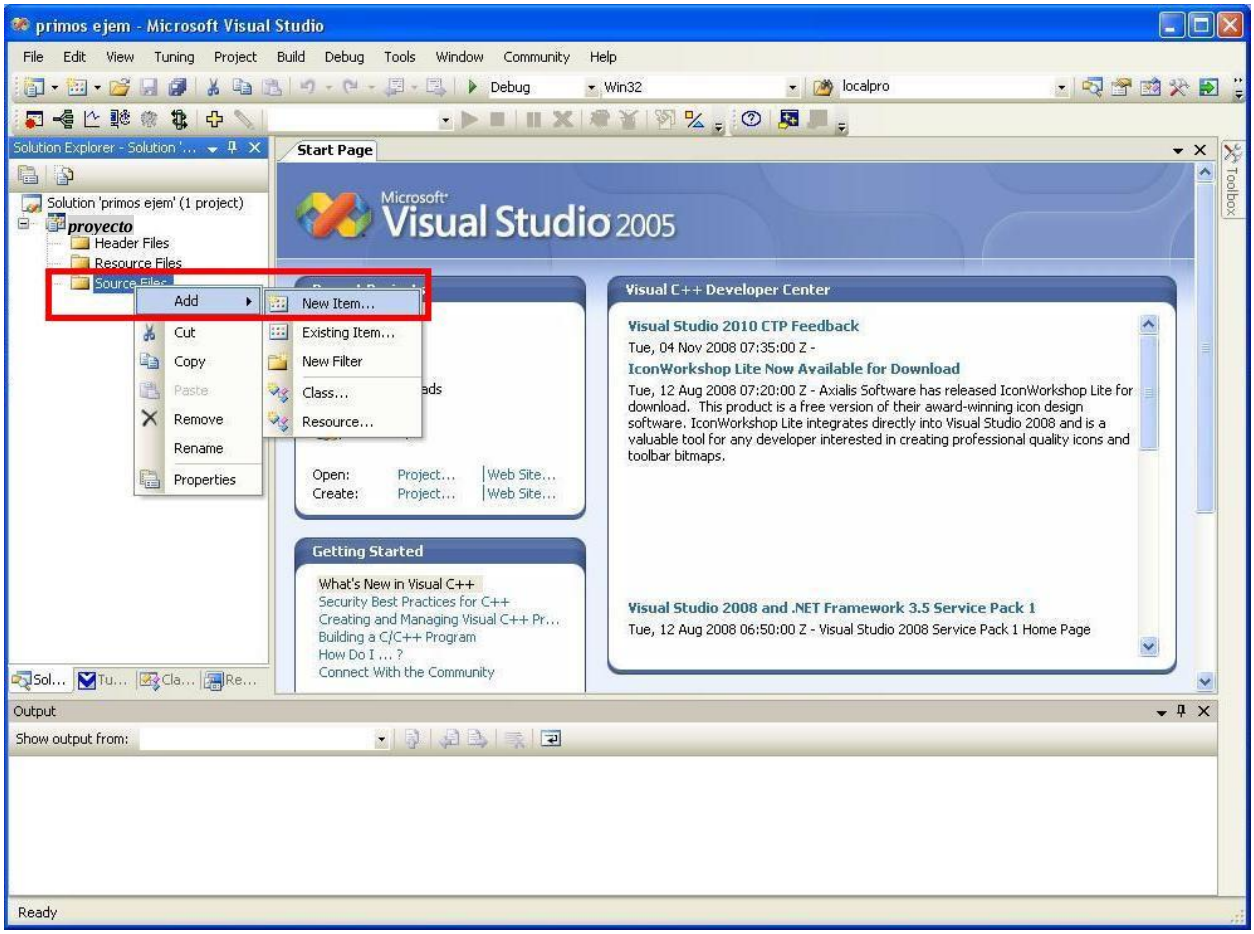
3.- Seleccionar el botón *Next*



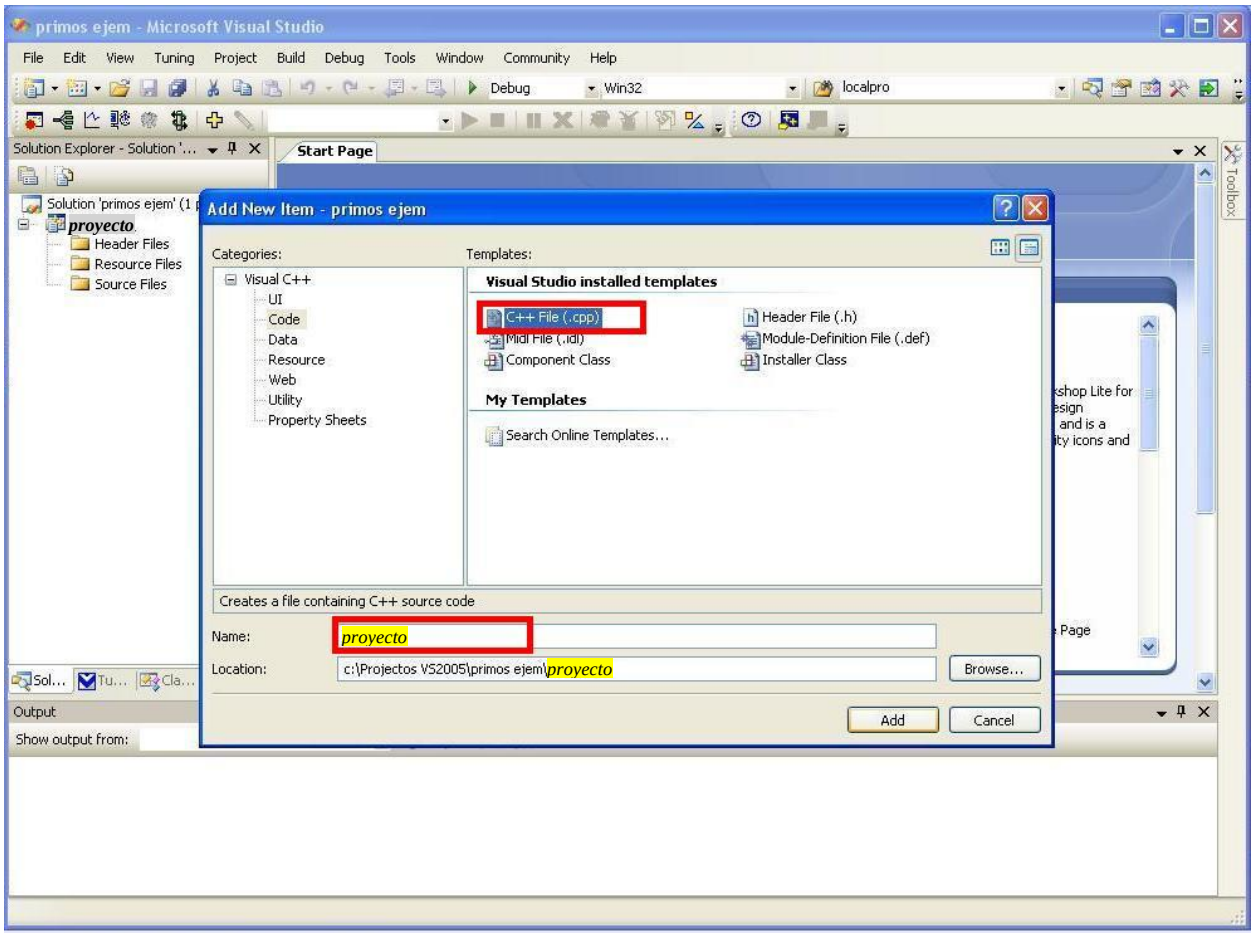
4.- Seleccionar *Console Application* y *Empty Project*, terminar presionando el botón *Finish*.



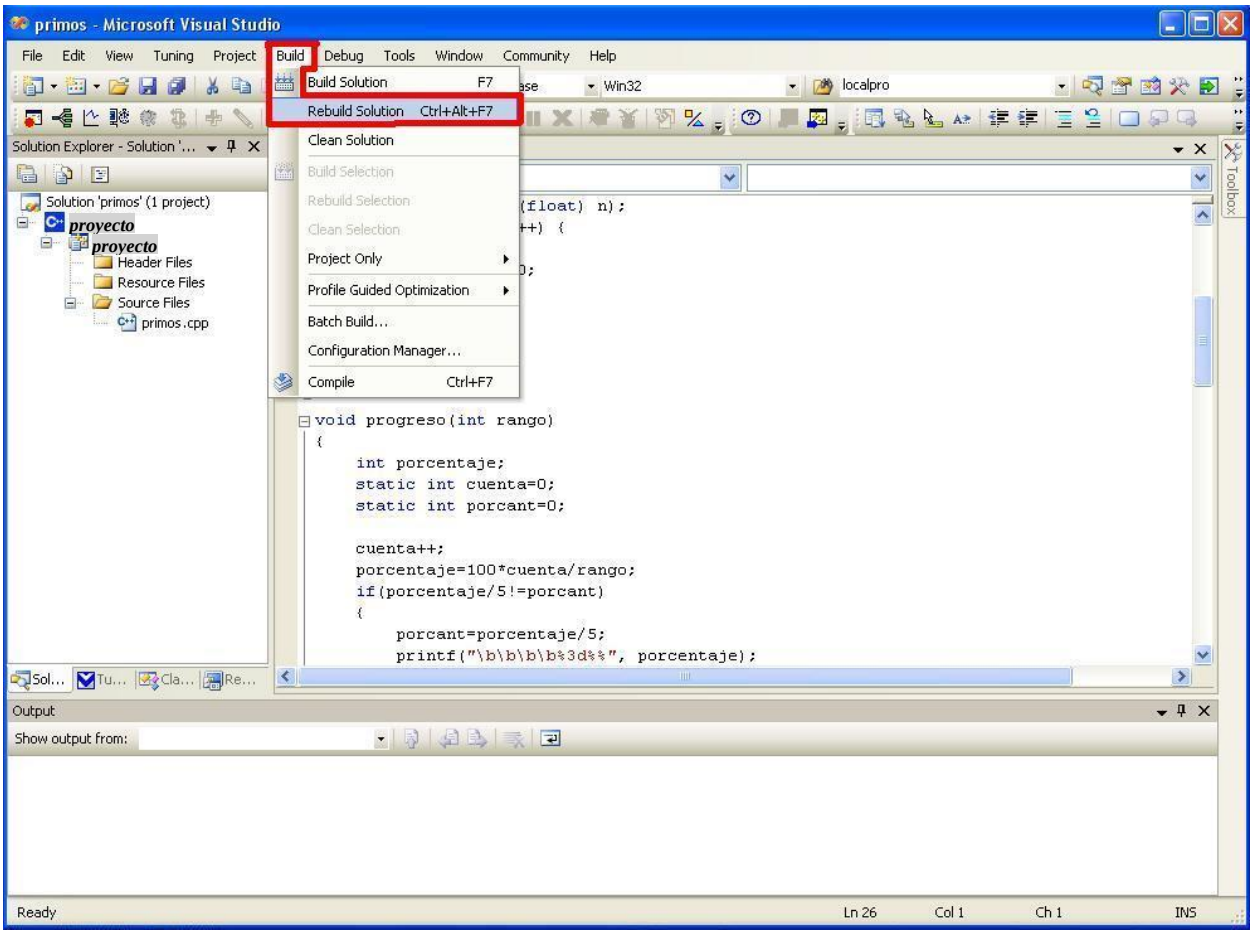
5.- Añadir el archivo fuente. Seleccionar *Source Files* y presionar el botón derecho del ratón. En el menú que se muestra a continuación seleccionar *Add* y posteriormente *New Item*.



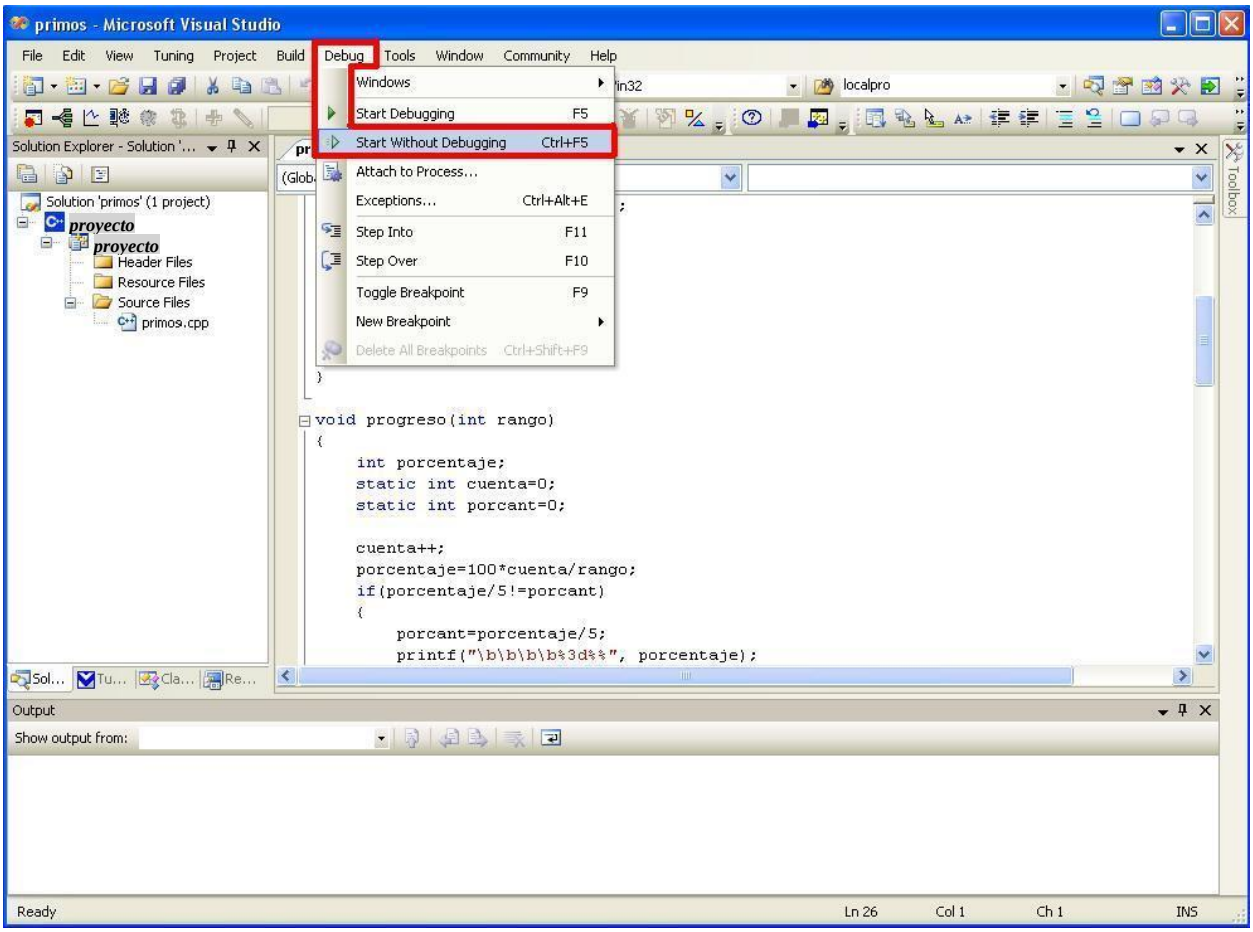
6.- Seleccionar archivo: *C++ File (.cpp)* y en la casilla *Name* nombrar el archivo



7.- Para compilar puede hacerlo presionando *<Ctrl><Alt><F7>* o seleccionar en los menús *Build -> Rebuild Solution*.



8.- Para ejecutarlo puede hacerlo presionando *<Ctrl><F5>* o desde el menú *Debug->Start Without Debugging*



2.2 Encontrando el cómputo intensivo

Utilizando la herramienta Parallel Amplifier identifica las regiones que hacen uso intensivo del CPU.

1. ¿Cuáles son las líneas de código que demandan mayor tiempo del CPU?

2. Basándonos en la información que obtenemos en la pregunta anterior, ¿cuál es la mejor forma de paralelizar?

3. Basándonos en la ley de Amdahl y el reporte que nos muestra Parallel Amplifier, ¿Cuál es la aceleración esperada a lograr con 2 y 4 procesadores?

2.3 Versión paralela del programa

Modifica el programa serial *detección de bordes*, de modo que agregando hilos de la API de Windows, este pueda sacar provecho de los procesadores Multicore.

La versión paralela deberá funcionar con dos, cuatro o *n* hilos solo modificando la constante `NHILOS` que se defina al inicio del programa, puede verse en el Ejemplo 2. Esto es para que fácilmente pueda ser re-compilado para sacar provecho de en un procesador de dos núcleos (Core Duo) o de cuatro núcleos (Quad Core).

Atención: No desarrolles dos versiones diferentes para diferente número de hilos, debes de usar funciones de hilos con parámetros.

```
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

#define NHILOS 2
```

Ejemplo 2. Constante que indica el número de hilos que creará el programa.

Ejecuta el programa en procesadores Core Solo, Core Duo y Quad Core, toma nota de los tiempos en la Tabla 1.

2.4 Tiempos de ejecución.

Basándote en las actividades previamente realizadas compara los tiempos de ejecución.

Procesador	Tiempo antes de Paralelizar	Tiempo después de Paralelizar
Single Core		
Dual Core		
Quad Core		

Tabla 1.- Comparación de rendimiento de las aplicaciones en diferentes procesadores antes y después de paralelizar.

V EVALUACIÓN

1 Equipos

Esta práctica se hará en equipos (máximo 2 integrantes), es necesario que en la revisión esté el equipo completo ya que el integrante que no se presente no tendrá calificación en la práctica.

2 Entrega

Fecha límite:

Miércoles 18 de Febrero hasta las 23:59 hrs.

- Un archivo .ZIP que contenga
  - El programa fuente (.c) con la versión paralela que se pide en la parte 2.3.
  - Un documento PDF con los resultados de experimentos que se piden en la parte 2.4.

Viernes 20 de Febrero hasta las 23:59 hrs.

- Un documento PDF con los reportes de los resultados de Parallel Amplifier

No incluya líneas de código en sus programas de las cuales desconozca su funcionamiento. El código no conocido será anulado en el funcionamiento de la práctica.

3 Evaluación

Puntualidad en las revisiones	El equipo estuvo completo y puntual en todas las sesiones de revisión.	Si hubo dos o más sesiones con el equipo, el equipo estuvo completo y puntual en casi todas las sesiones de revisión	Si solo hubo una sesión de revisión, el equipo no estuvo completo o no fue puntual. Si fueron dos o más sesiones de revisión, en más de una sesión el equipo no estuvo completo o fue puntual
	+10	+5	0
Funcionamiento	El producto cumple con todas las especificaciones indicadas en el documento y no tiene fallas	El producto muestra una falla no esperada o el producto está casi completo, puede funcionar excepto la parte no completada	El producto muestra más de una falla inesperada o no funciona, esto puede ser debido a que no esté completo.
	+75	+37.5	0
Interfaz con el usuario	El producto funciona y pudo ser utilizado sin necesidad de recibir indicaciones por el desarrollador, tiene instrucciones claras para ser utilizado.	El producto funciona, pero hubo necesidad de recibir alguna indicación para su uso por parte del desarrollador del producto	El producto carece de instrucciones claras para ser utilizado y requiere que alguno de los desarrolladores esté presente para su utilización o no puede utilizarse debido a que no está completo
	+10	+5	0
Claridad en el código	El código es claro, usa nombres de variables adecuadas, está debidamente comentado e indentado. Puede ser entendido por cualquier otra persona que no intervino en su desarrollo.	El código carece de claridad, puede ser entendido por cualquier persona ajena a su desarrollo pero con cierta dificultad.	El código carece de comentarios, está mal indentado, usa nombres de variables no adecuadas.
	+5	+2.5	0
Defensa del producto	Todos los integrantes son capaces de explicar cualquier parte del producto presentado	Alguno de los integrantes muestra dudas sobre alguna parte del desarrollo del producto presentado	Más de un integrante, o si el trabajo fue individual , el desarrollador duda sobre cómo está desarrollado producto.
	x 1(puntos se multiplican por 1)	x 0.5 (puntos se multiplican por 0.5)	x 0 (puntos se multiplican por 0)
Sobresaliente 20 %	Tiene 1 en todos los puntos anteriores. El producto entregado es sobresaliente, muestra tener la calidad para ser expuesto como un producto representativo de la carrera Hay evidencia de que los desarrolladores se documentaron y muestran aprendizajes más allá de lo esperado		No tiene 1 en todos los puntos anteriores, o el producto entregado no es sobresaliente y no muestra tener la calidad para ser expuesto como un producto representativo de la carrera o no hay evidencia de que los desarrolladores se documentaron y muestran aprendizajes más allá de lo esperado
	+20		0