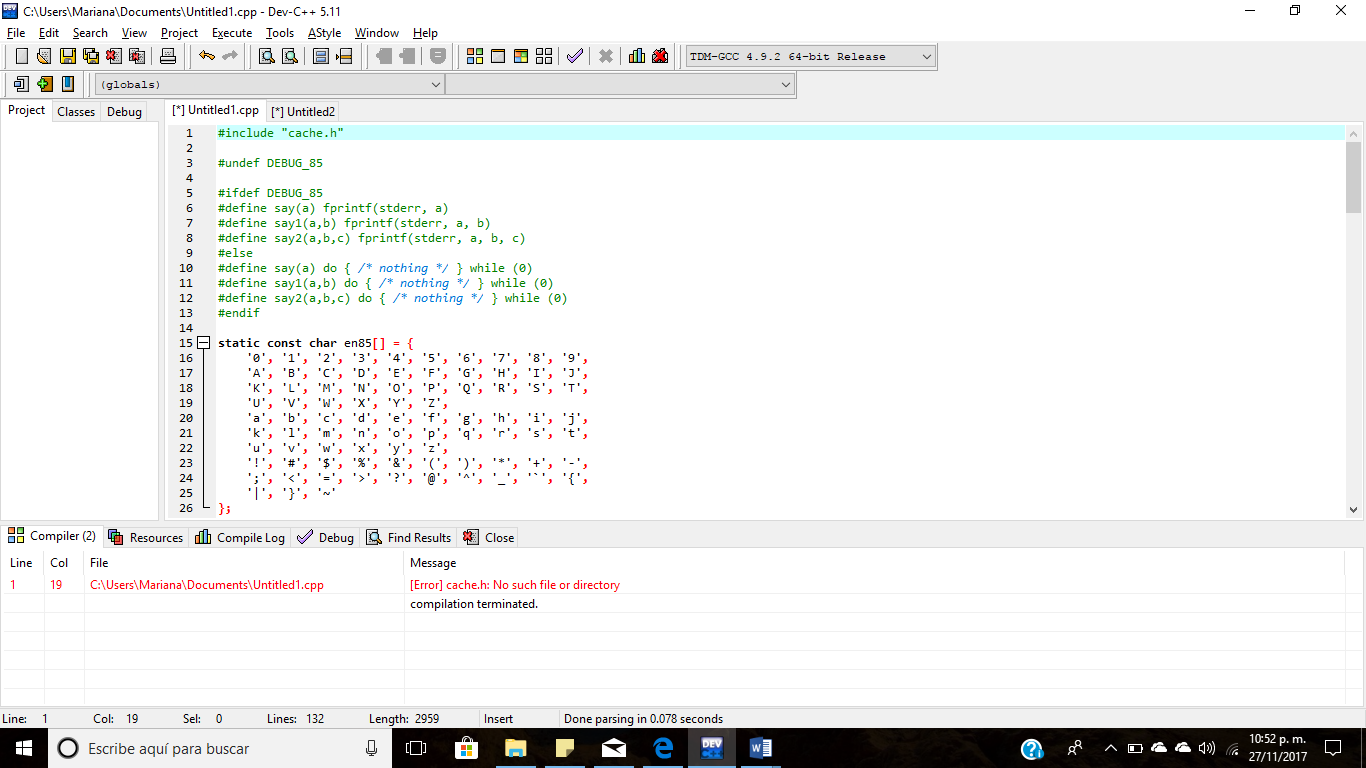
|  |  |  |
| --- | --- | --- |
|  | **Carátula para entrega de prácticas** | |
| Facultad de Ingeniería | | Laboratorio de docencia |

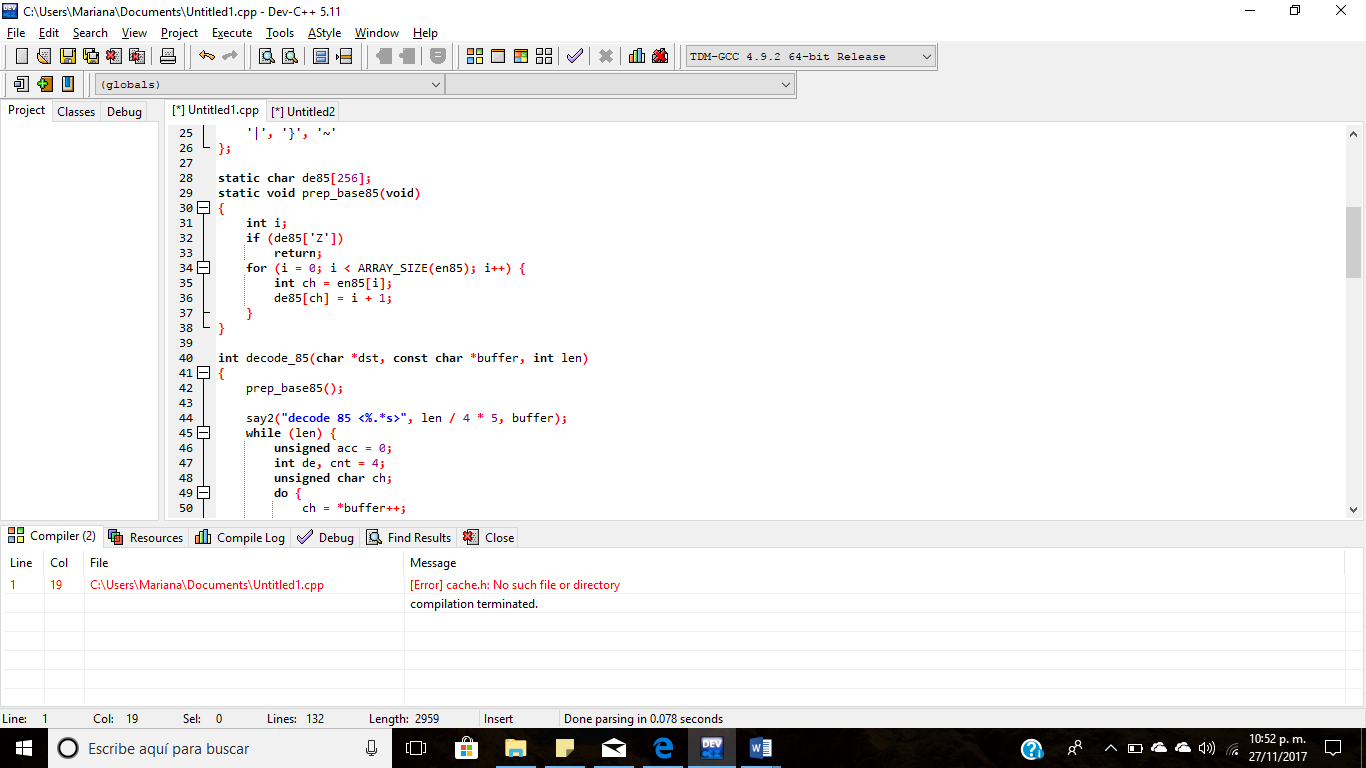
Laboratorios de computación

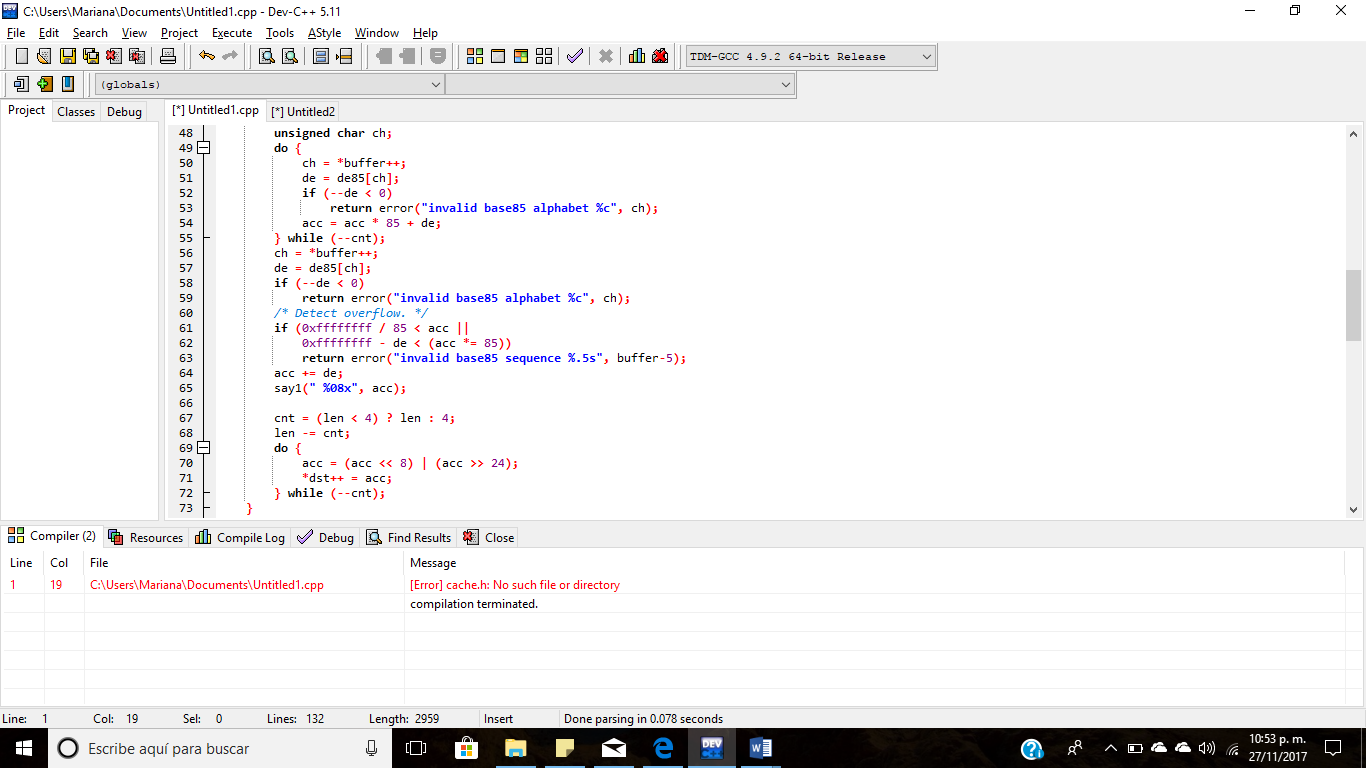
salas A y B

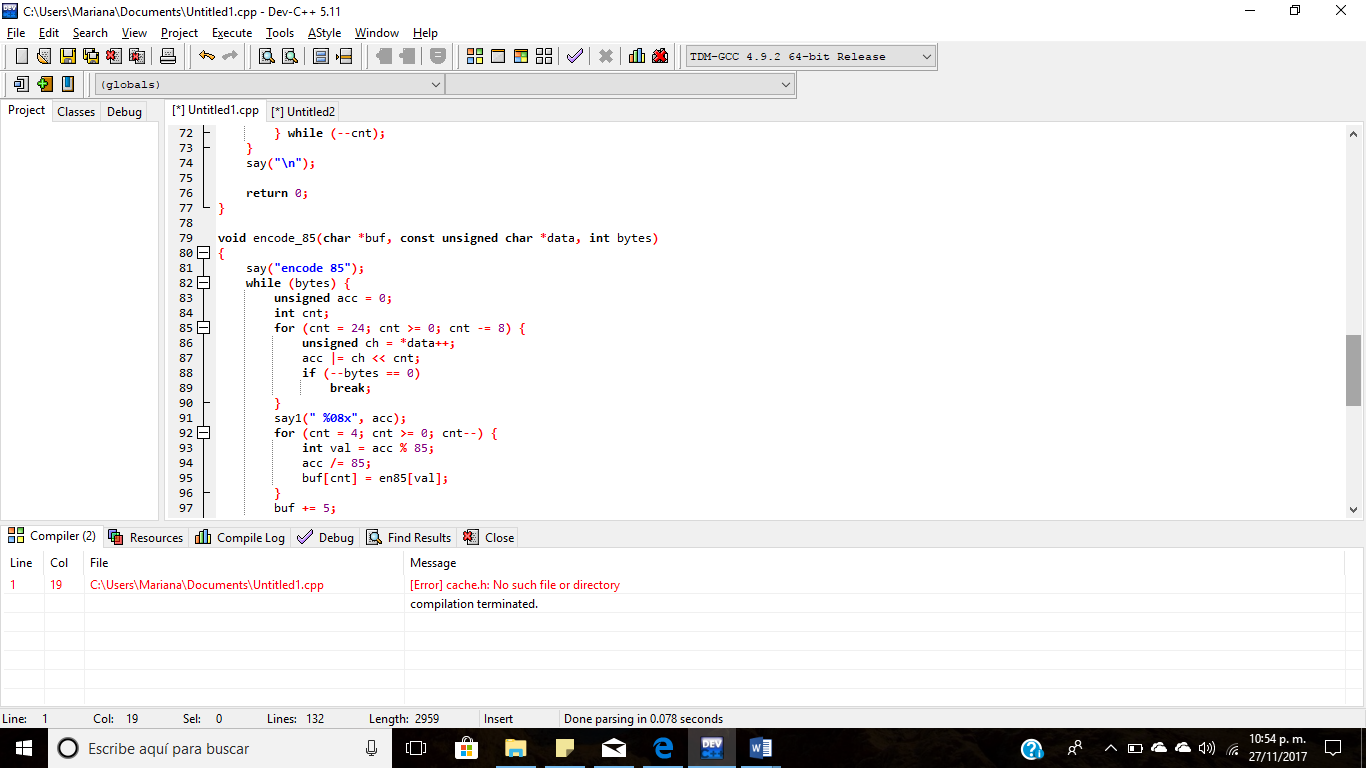
|  |  |
| --- | --- |
| *Profesor:* | Juan Alfredo Cruz Carlón |
| *Asignatura:* | Fundamentos de programación |
| *Grupo:* | 1107 |
| *No de Práctica(s):* | Práctica 12 |
| *Integrante(s):* | Hernández Ramírez Angel |
|  | Ramírez Robles Jeyelly |
|  | Rodríguez Ruíz Mariana |
| *Semestre:* | 2018-1 |
| *Fecha de entrega:* | 27 de noviembre de 2017 |
| *Obervaciones:* |  |
|  |  |

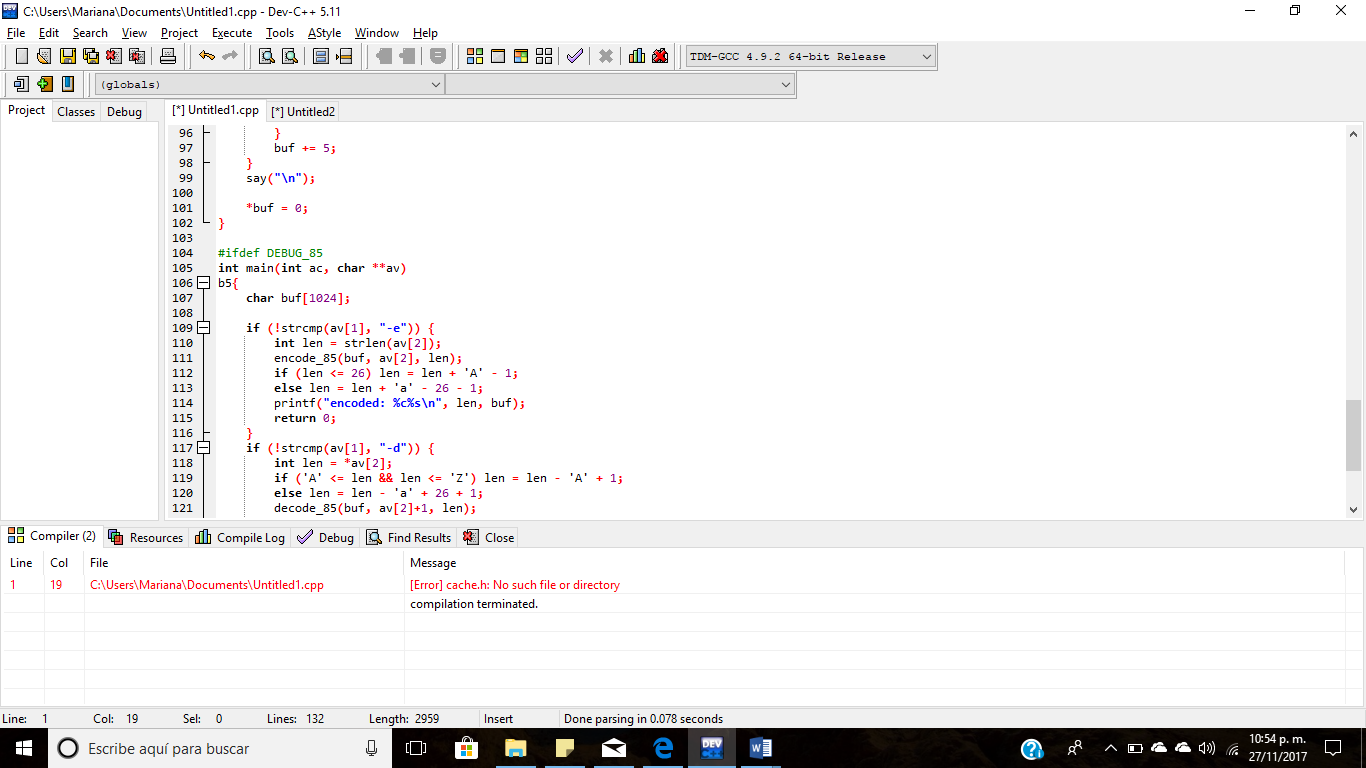
CALIFICACIÓN: \_\_\_\_\_\_\_\_\_\_

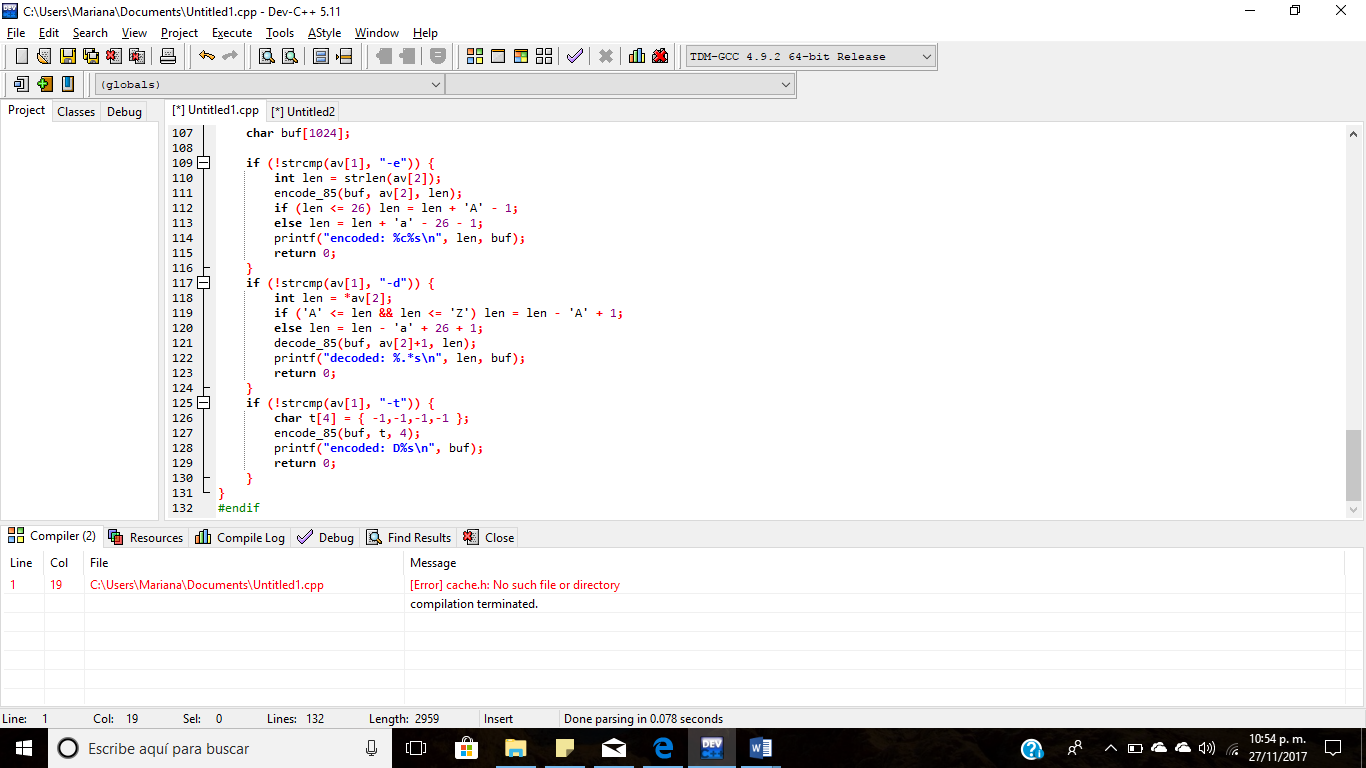


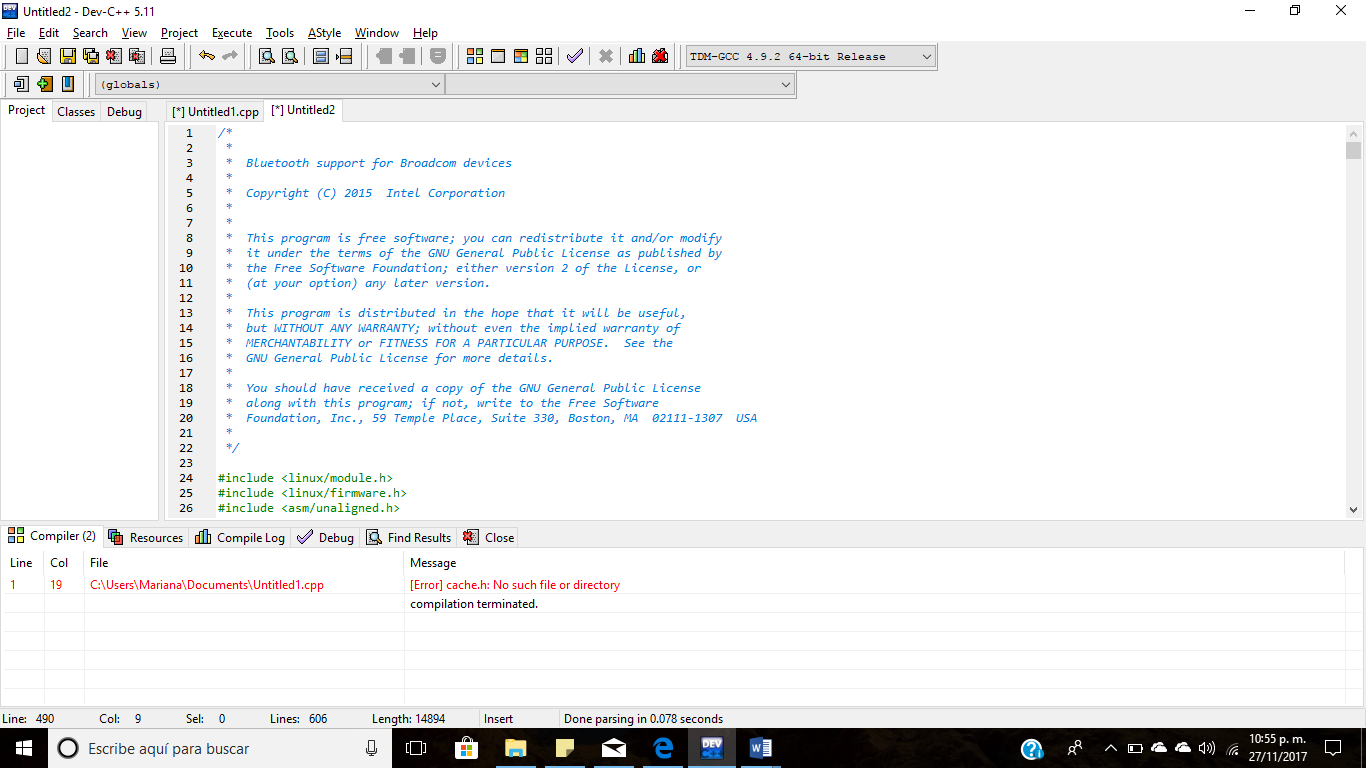


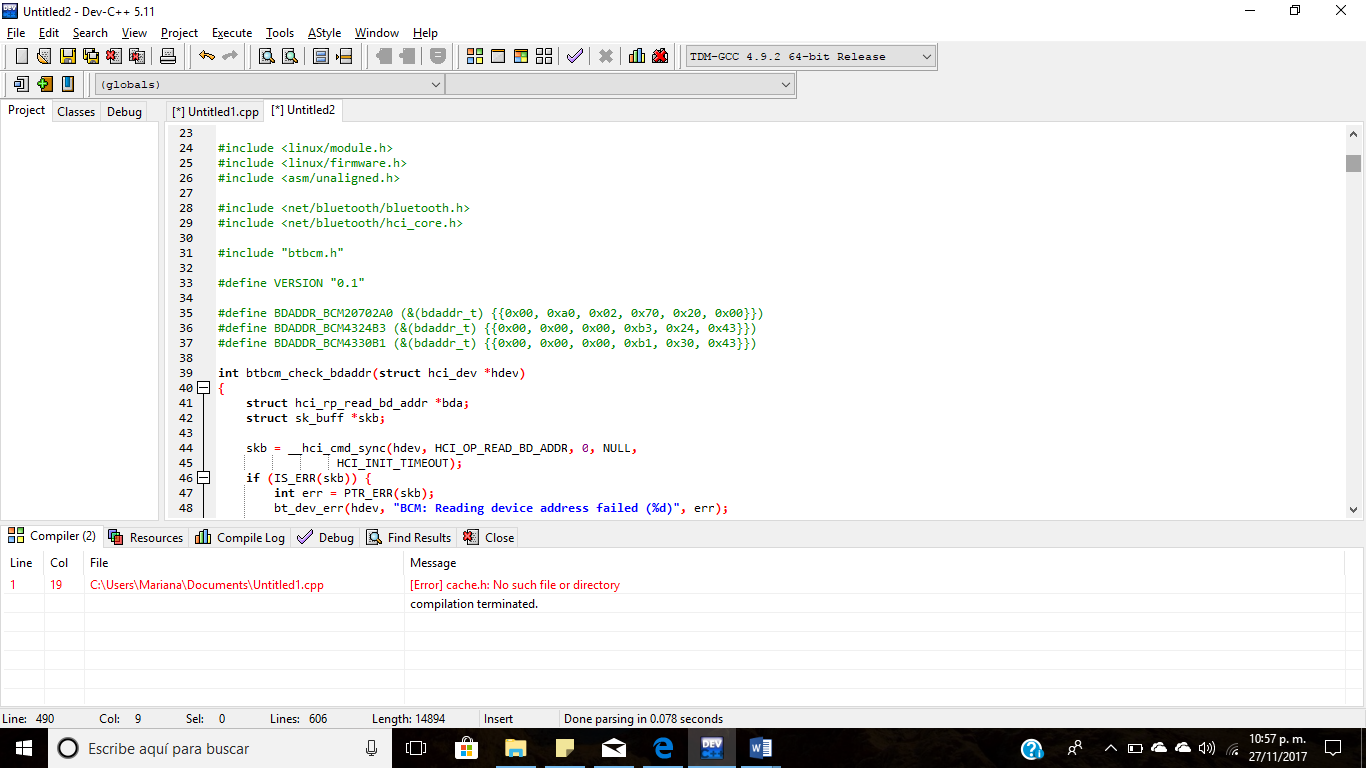


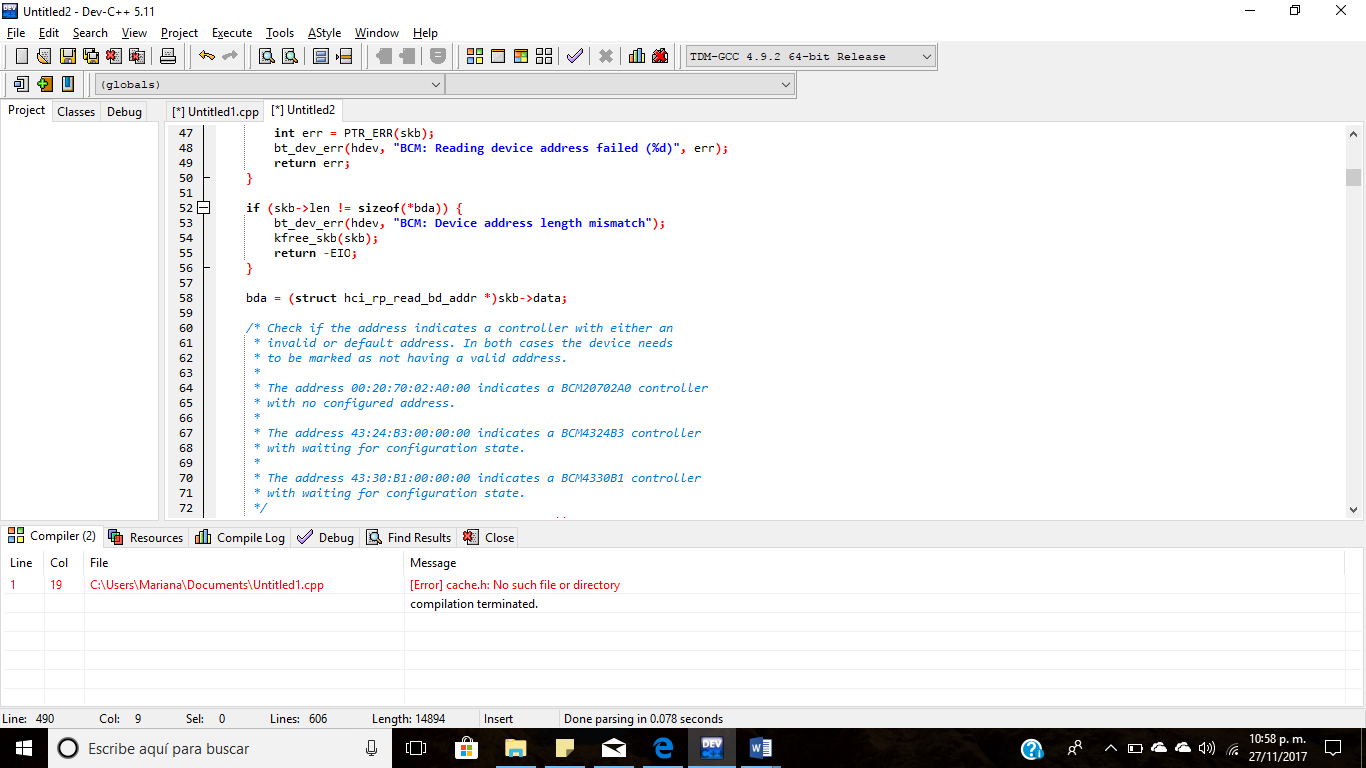


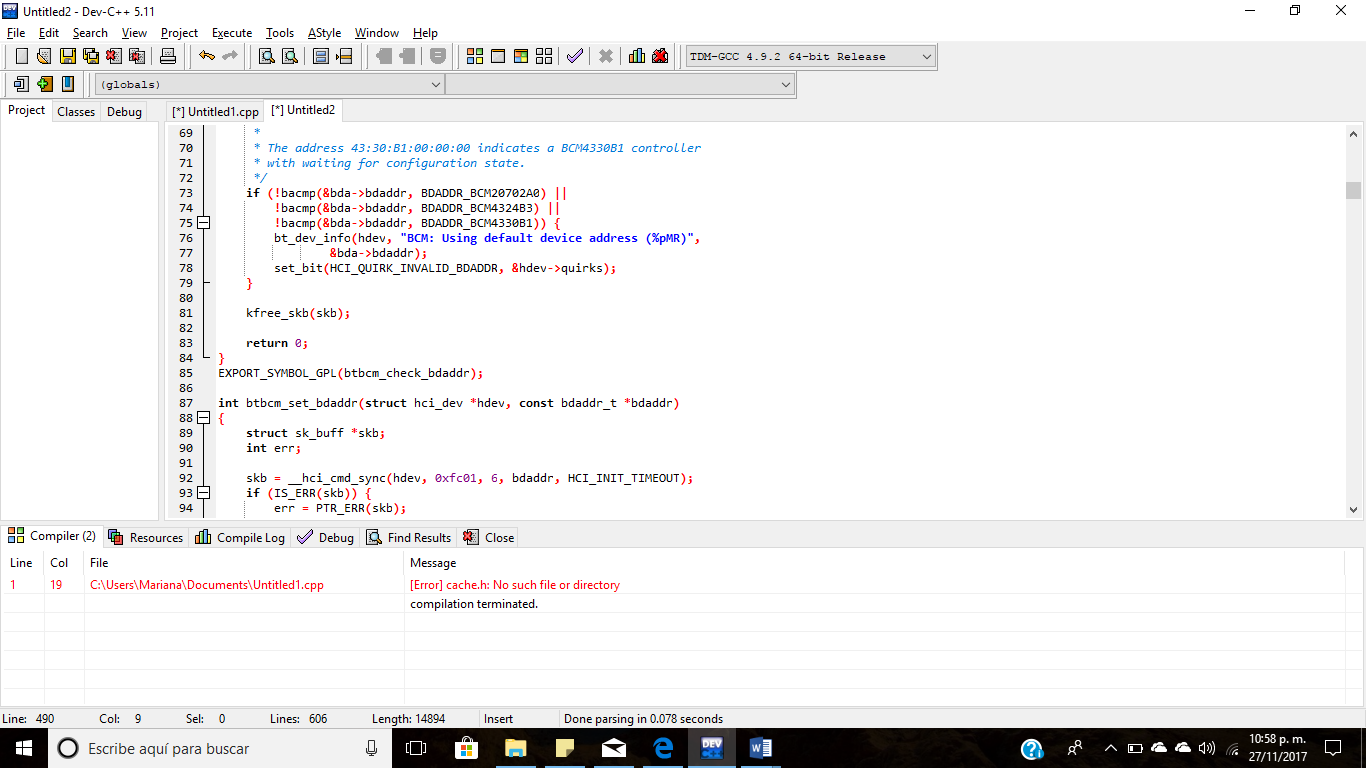


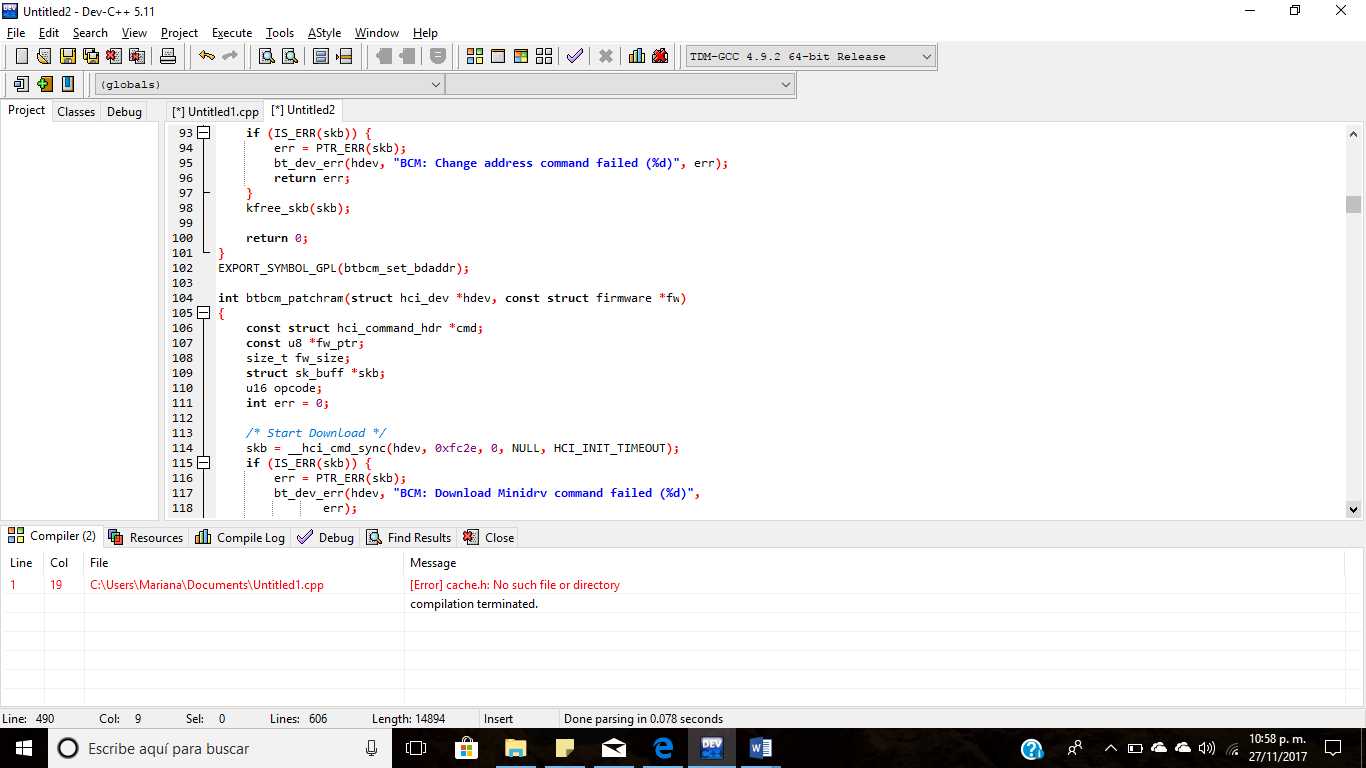


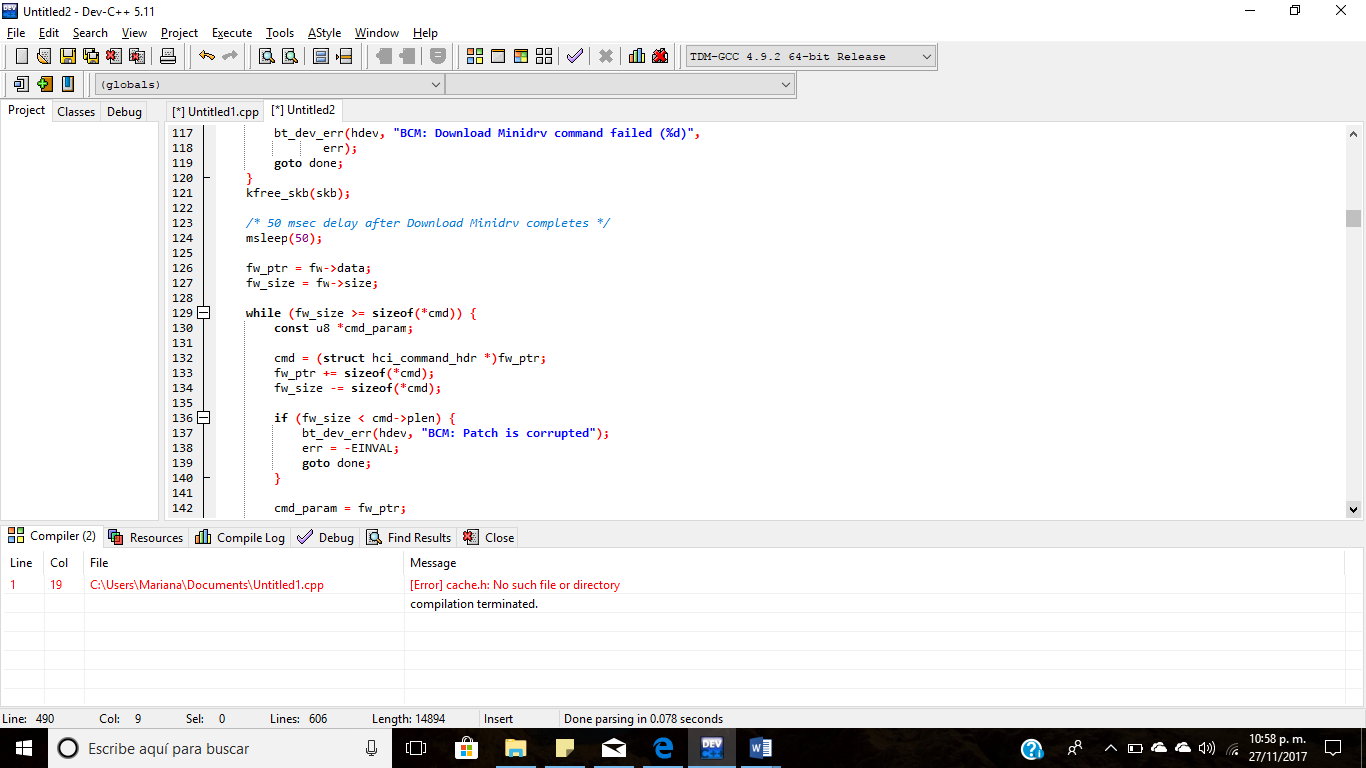


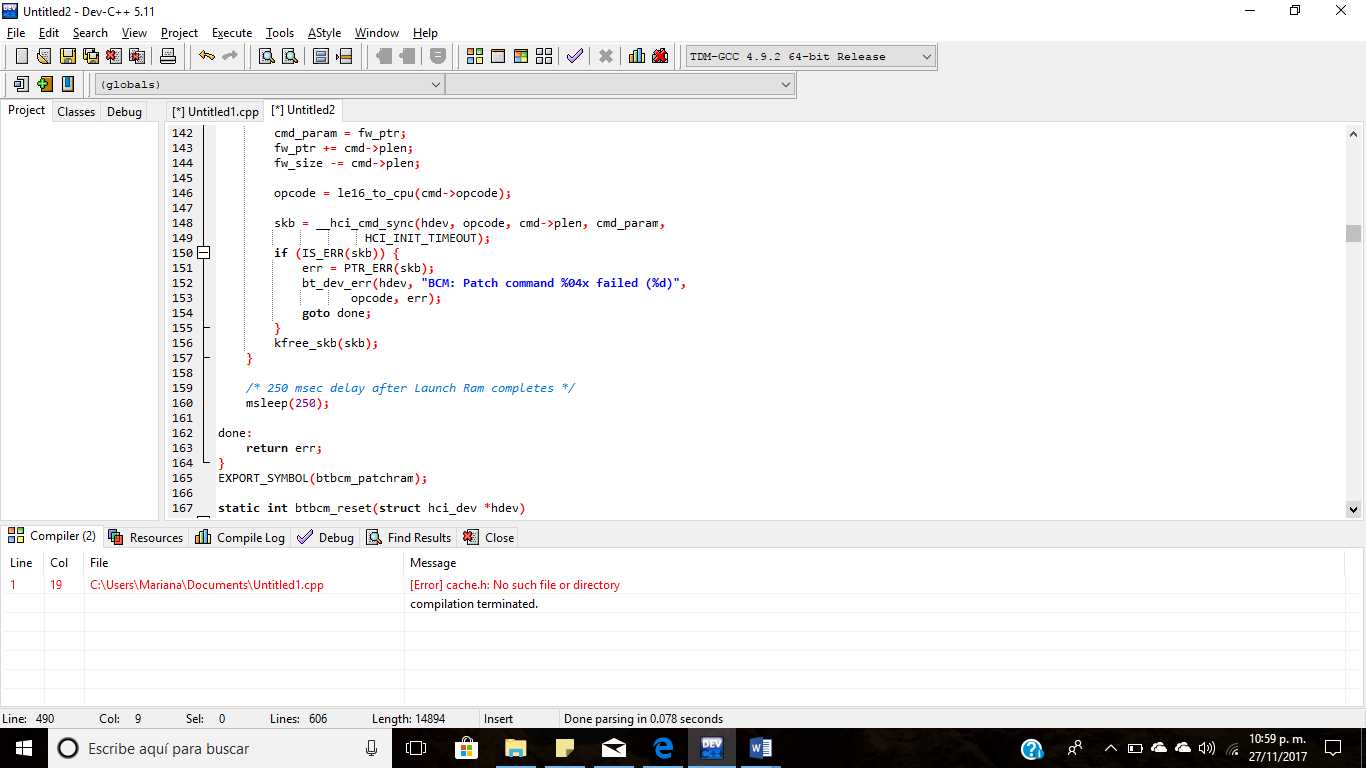


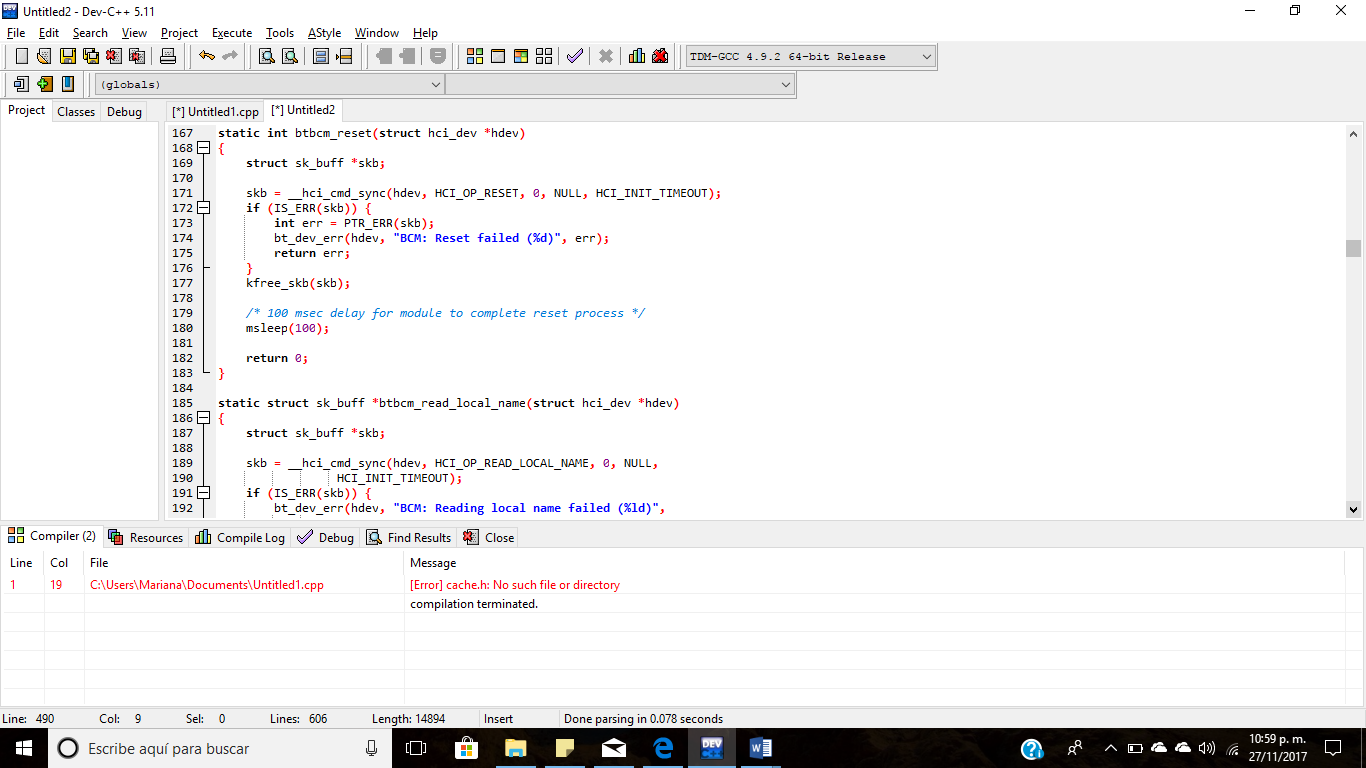


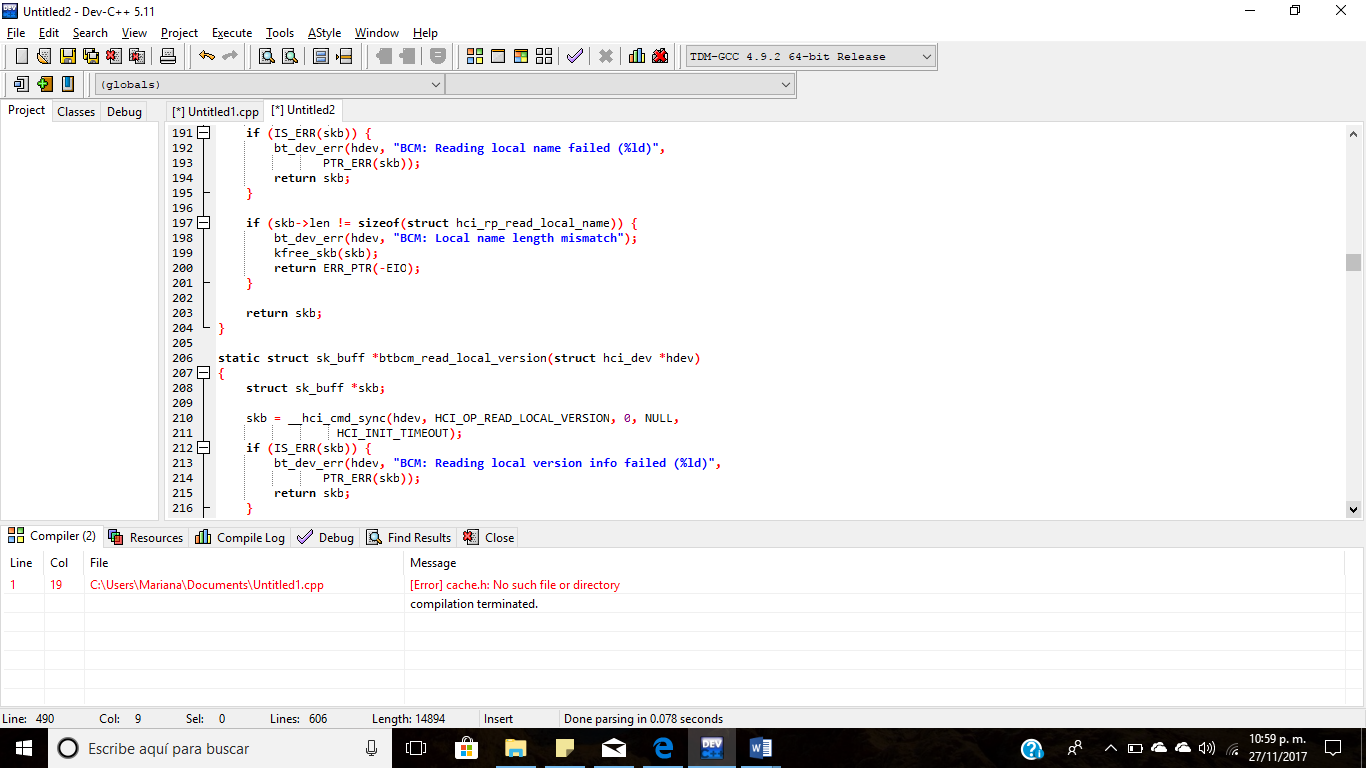


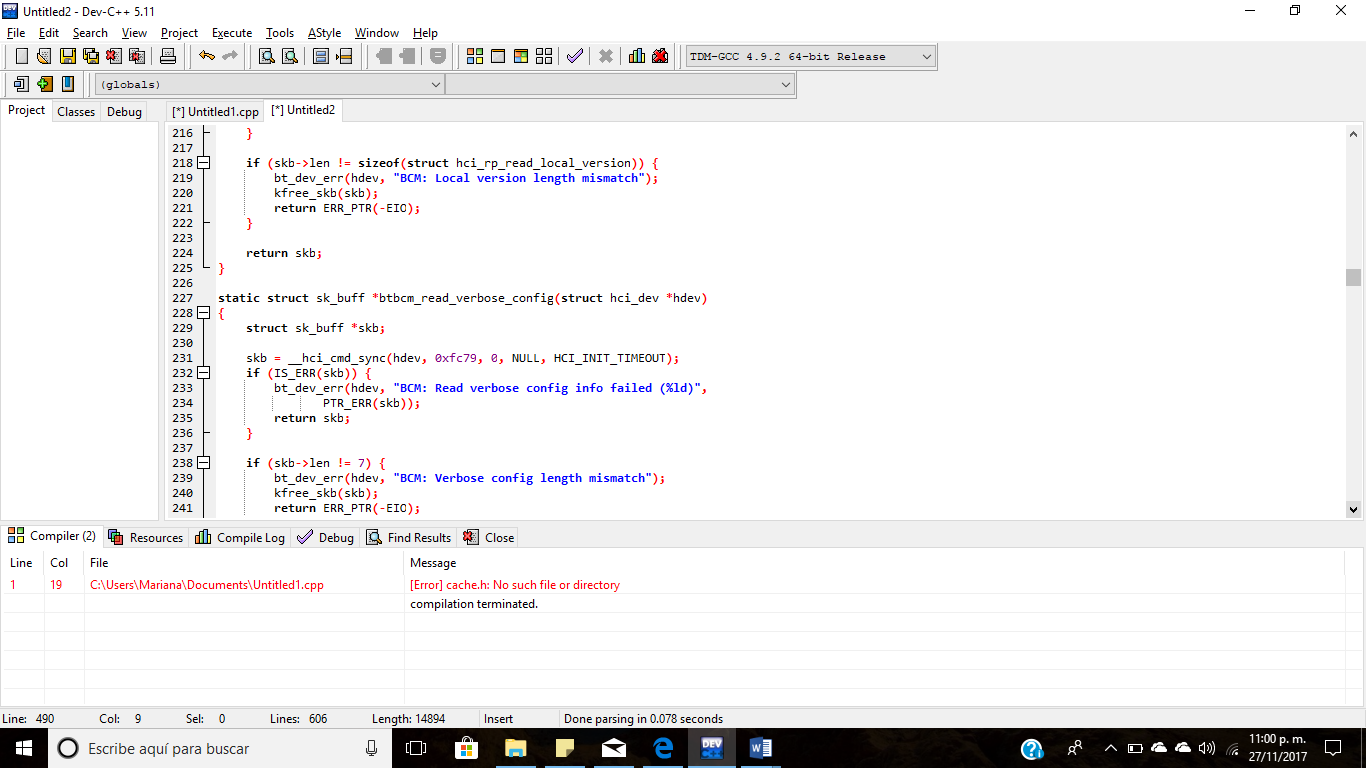


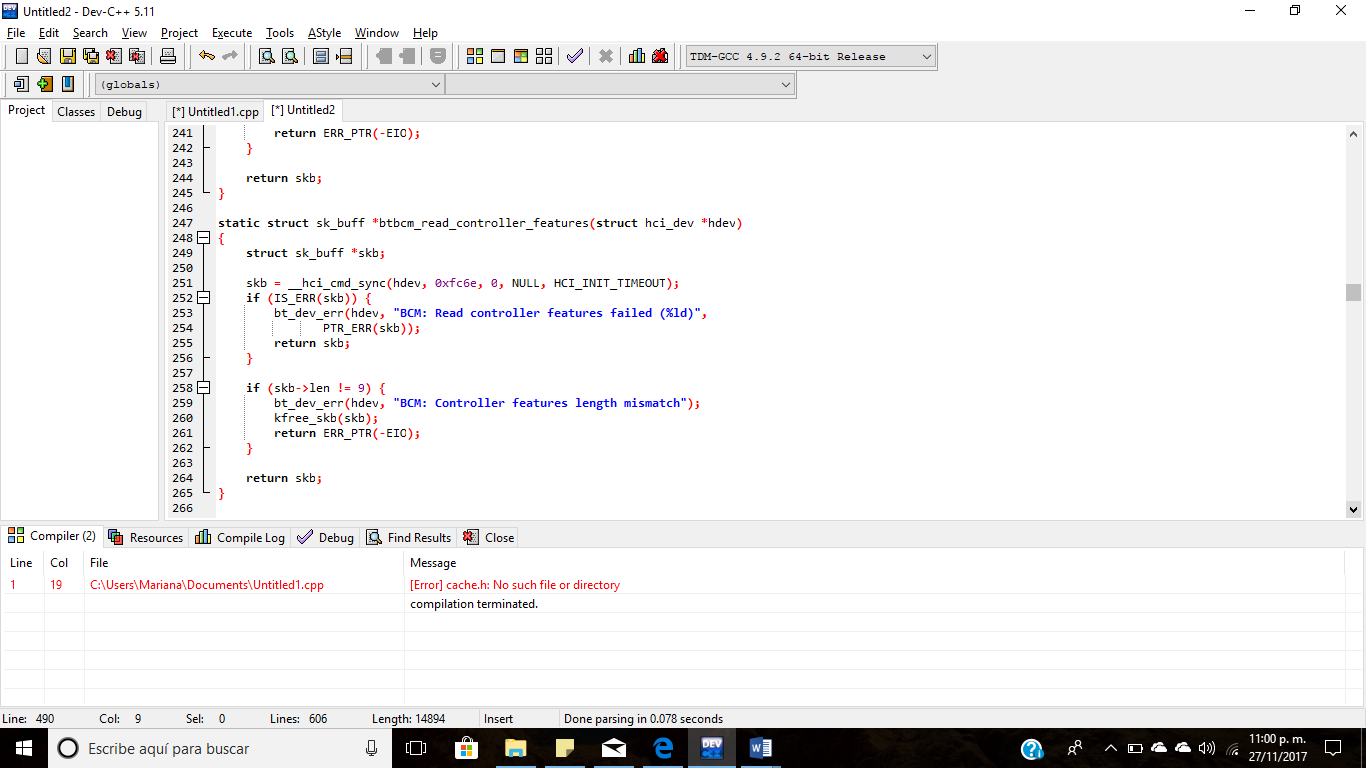


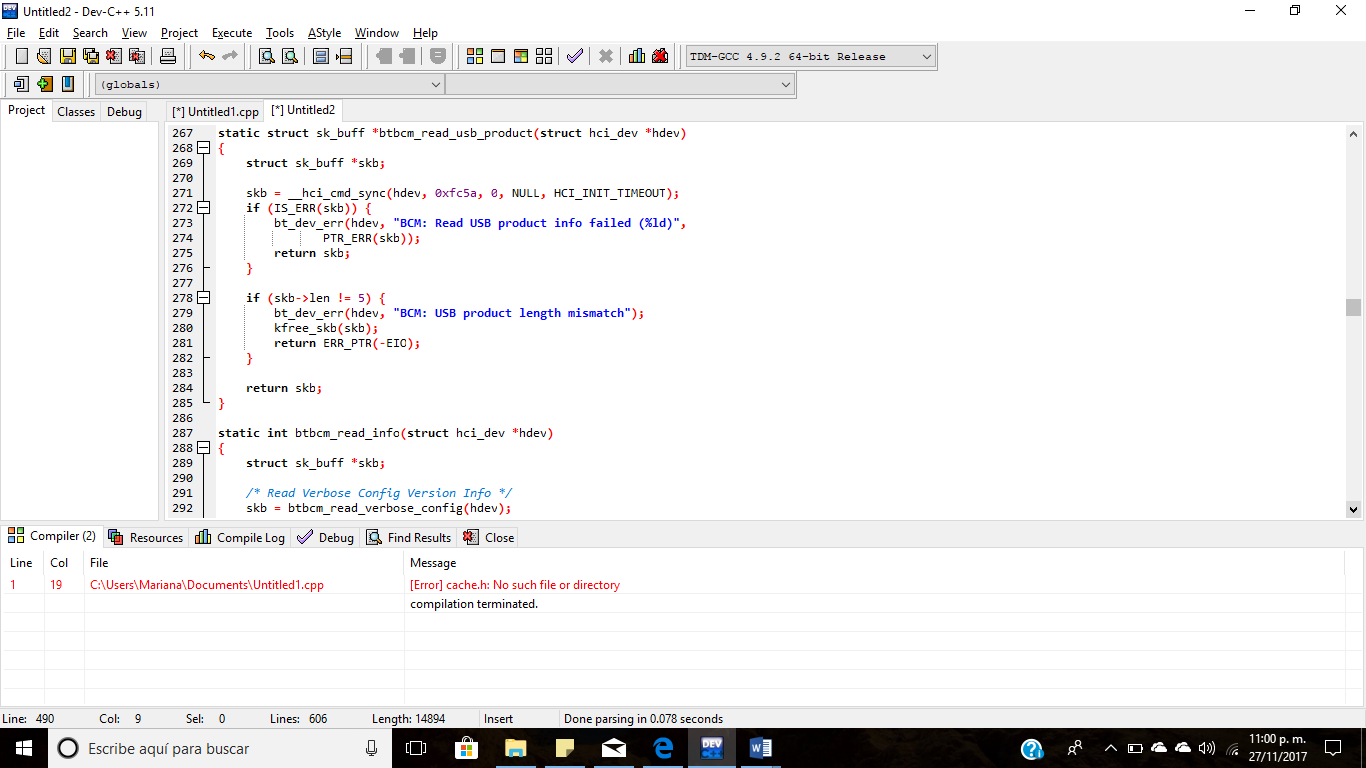


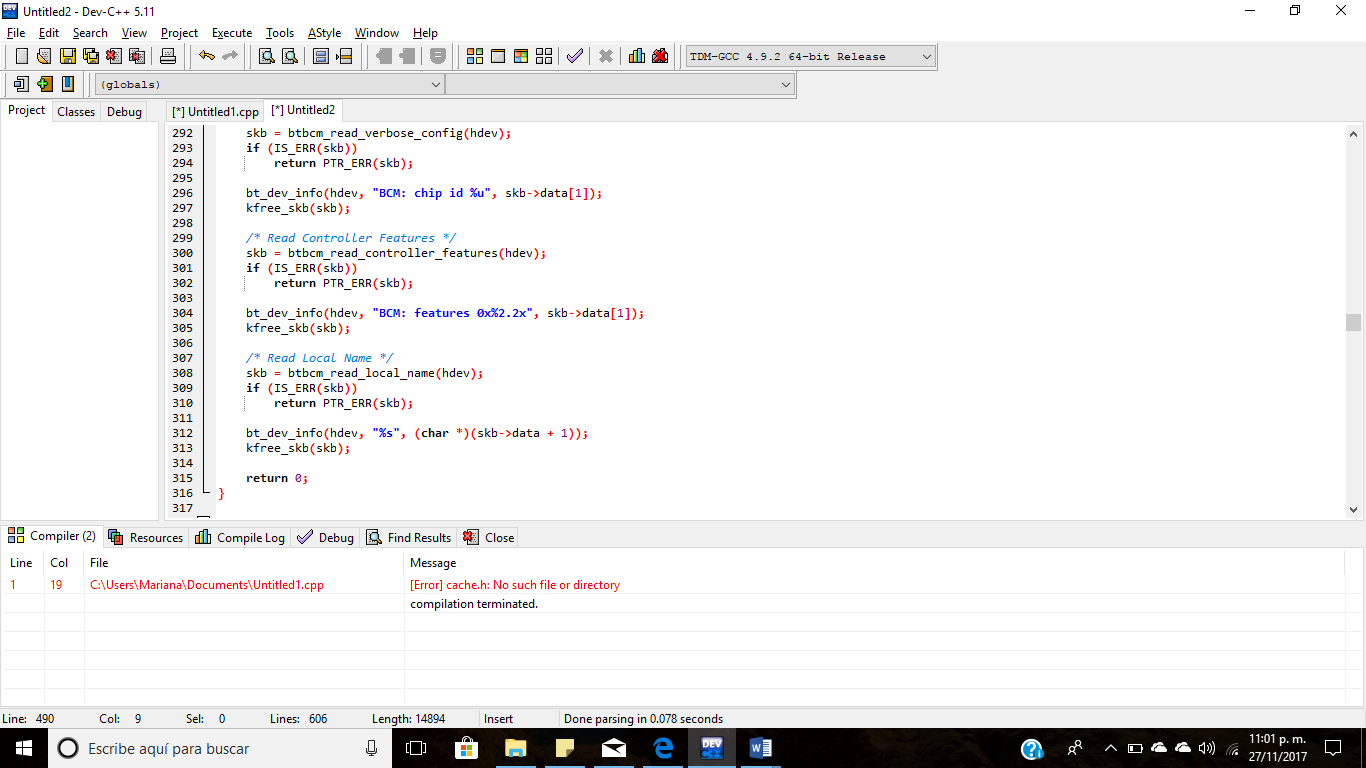


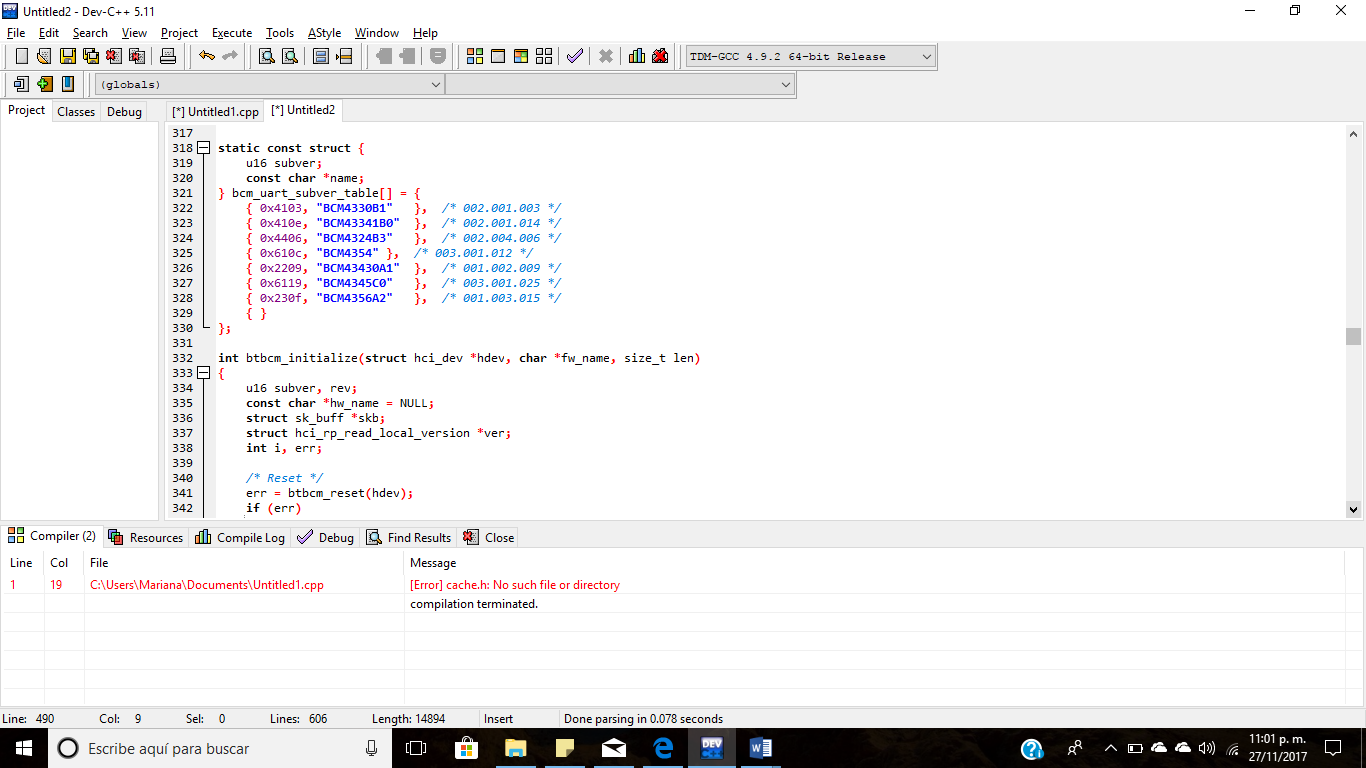


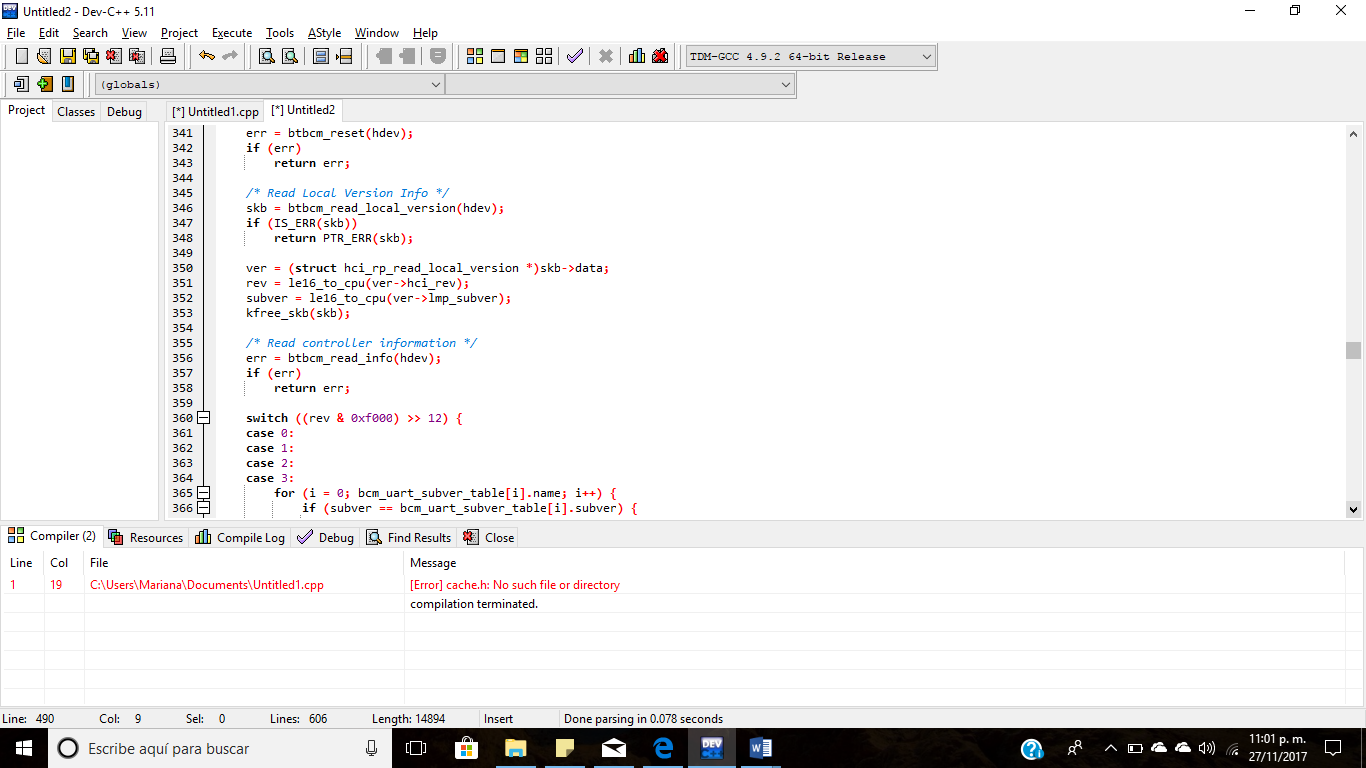


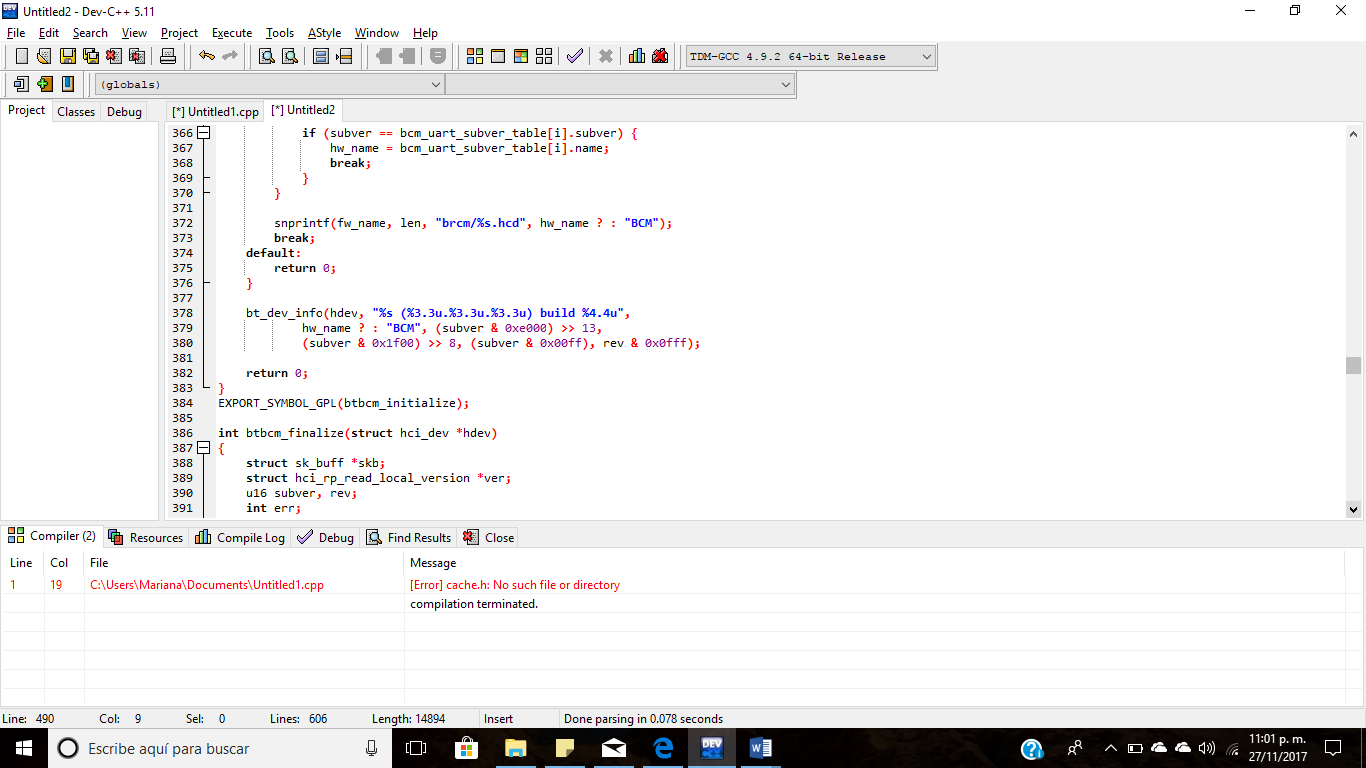


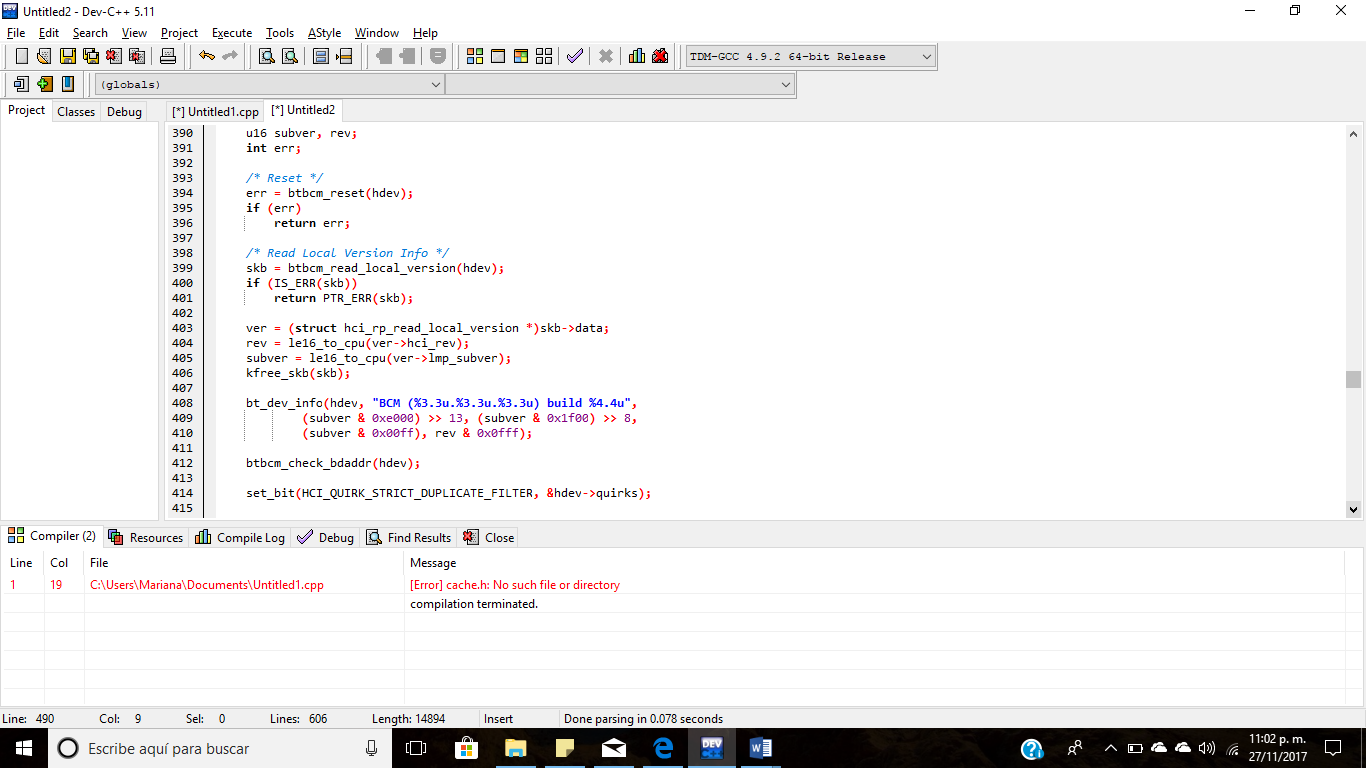


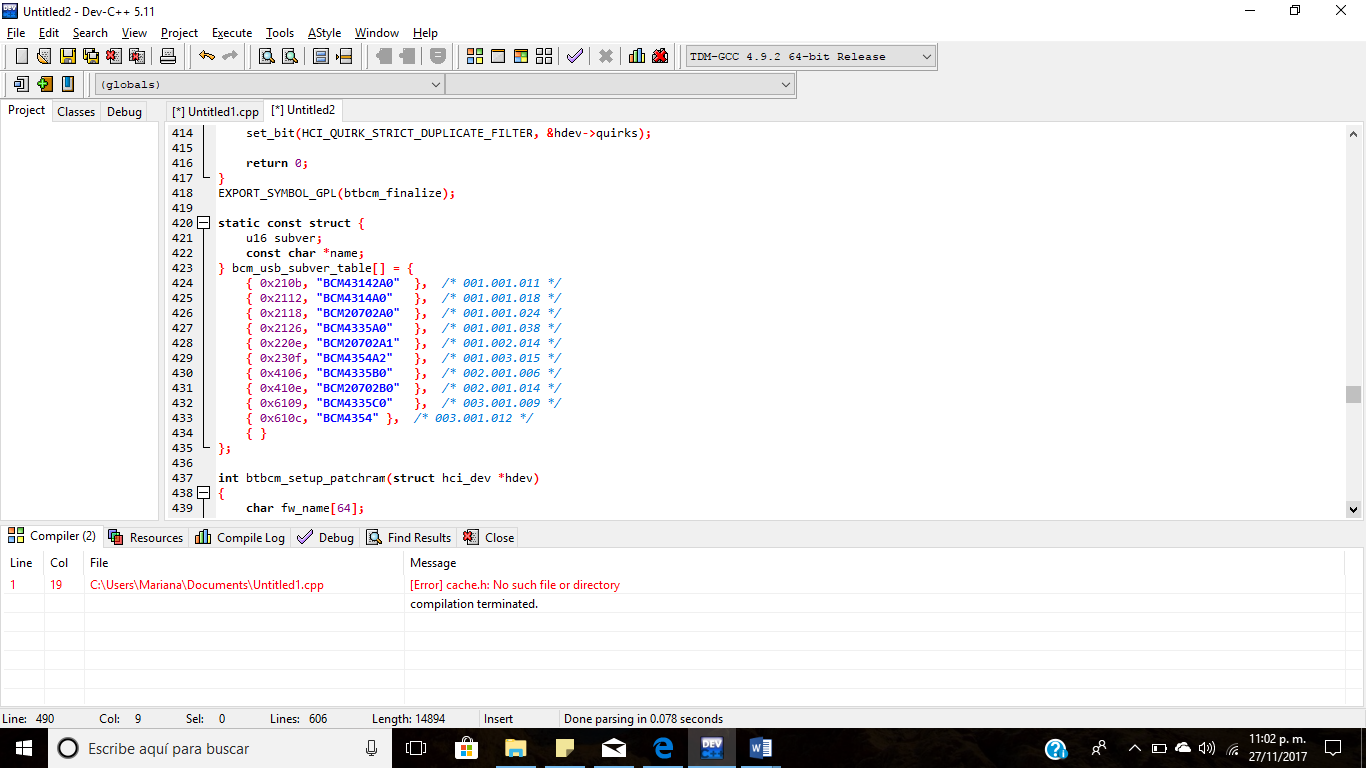


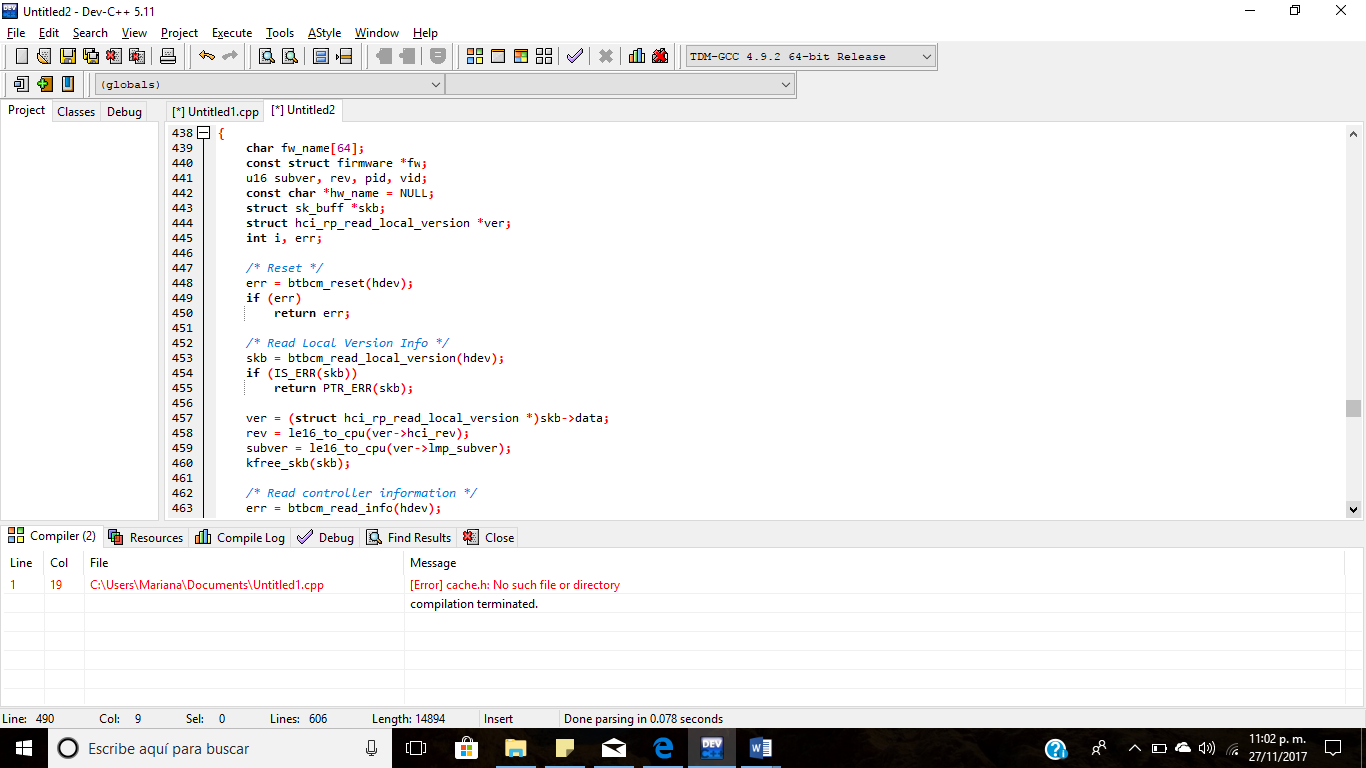


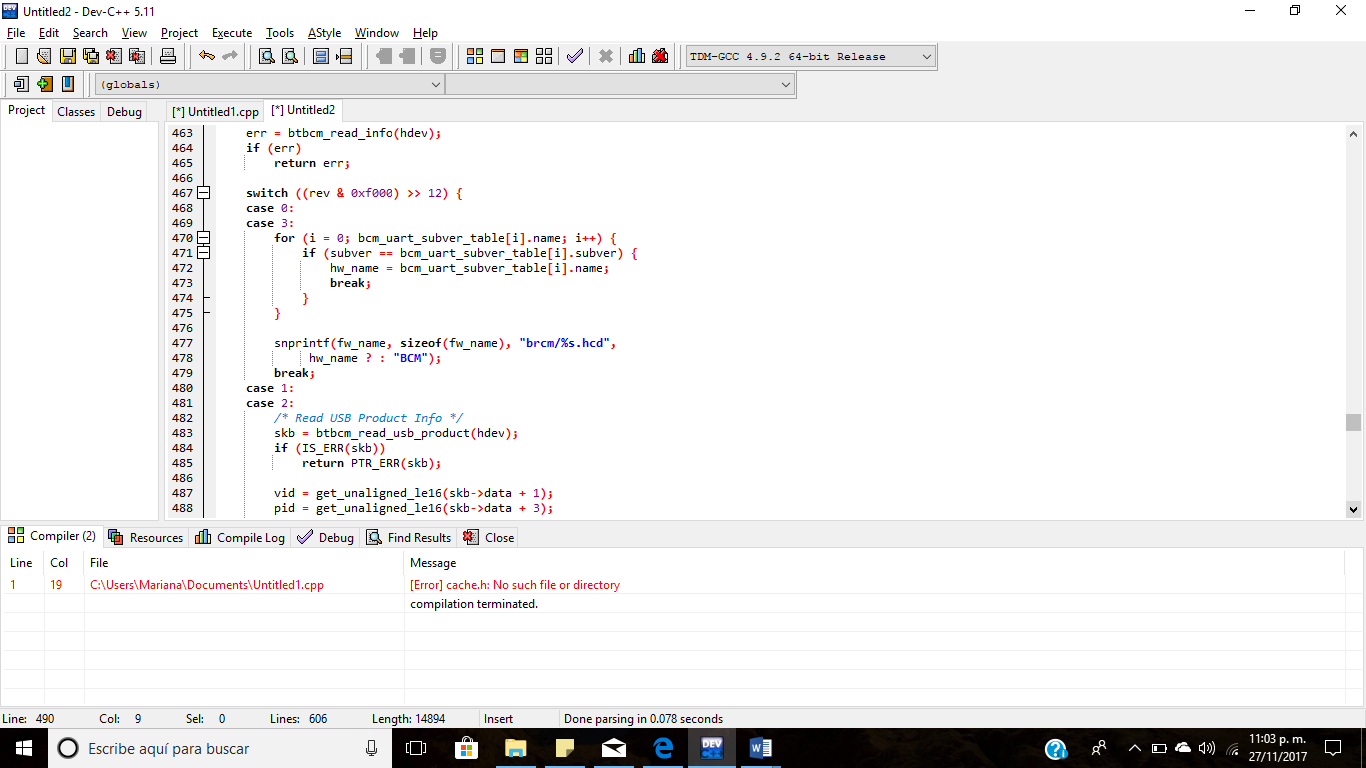


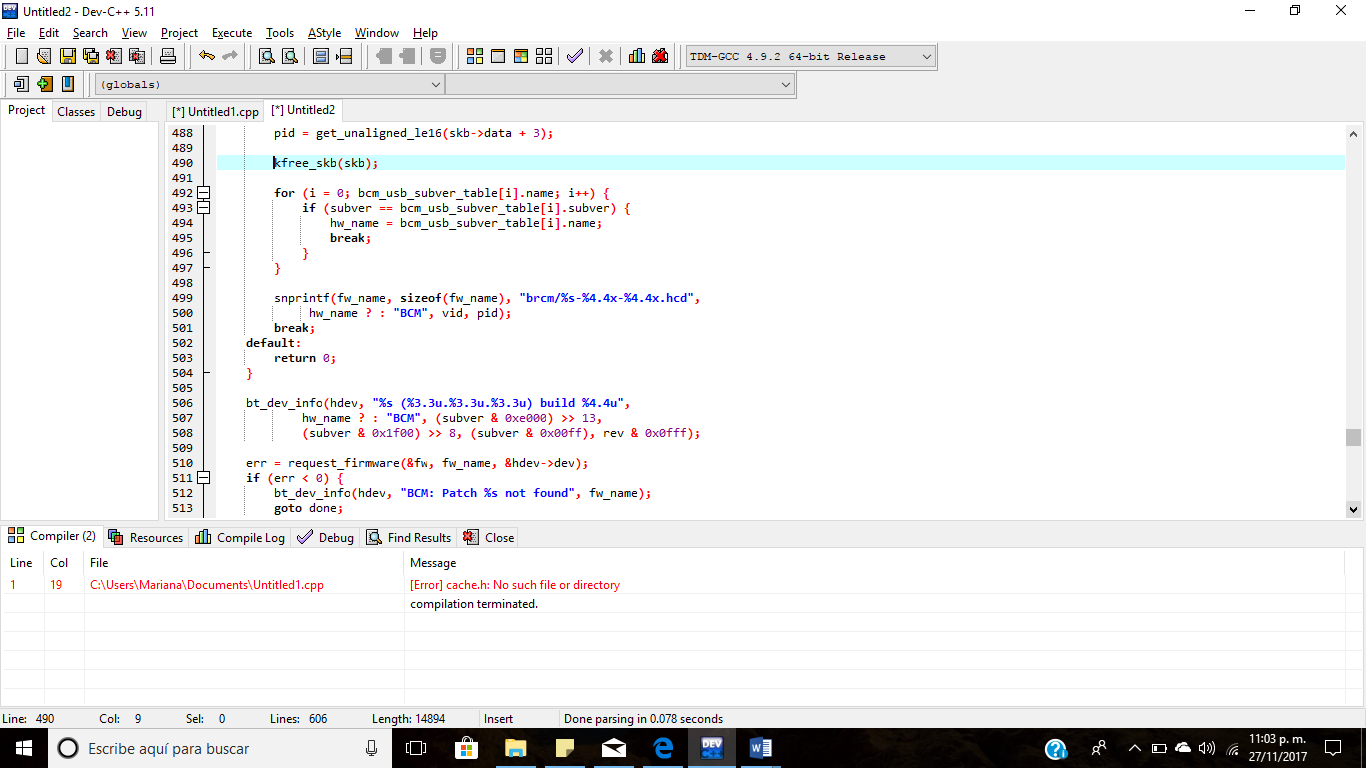


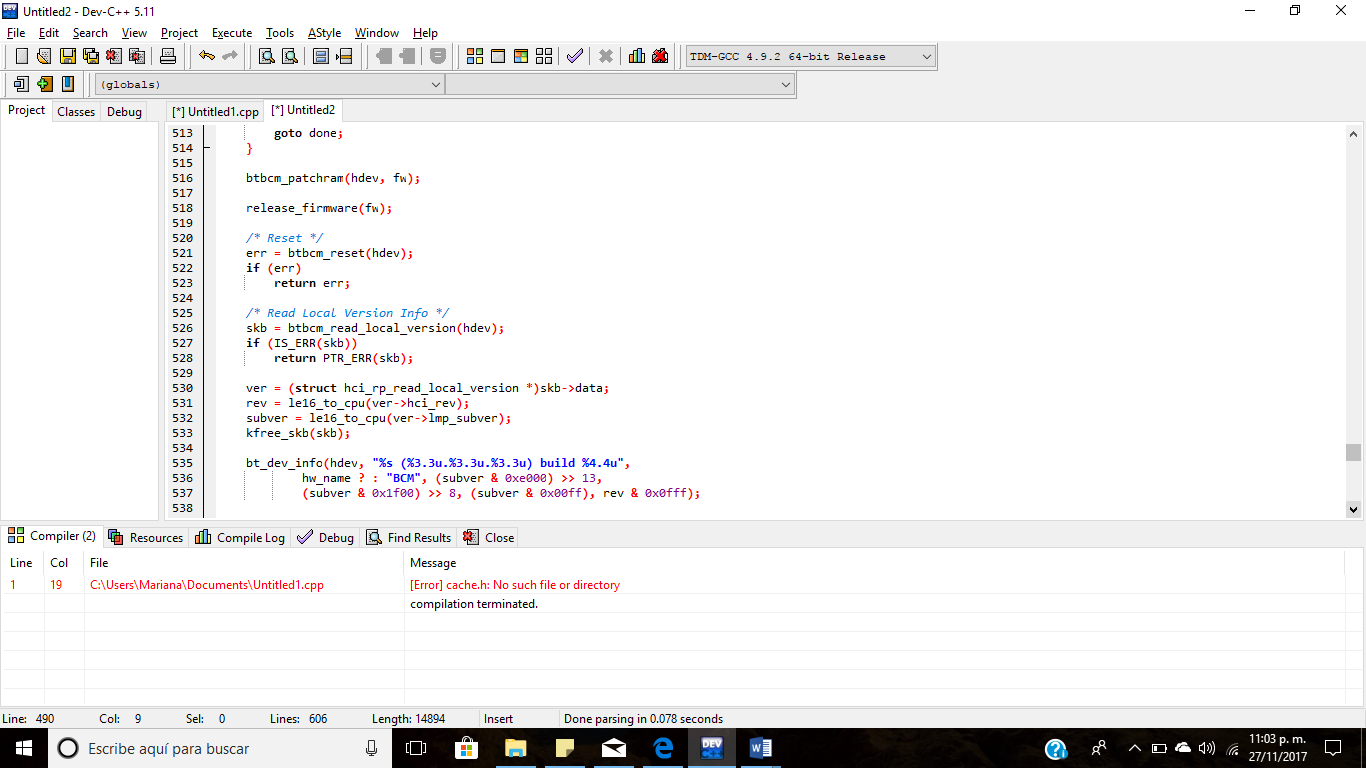


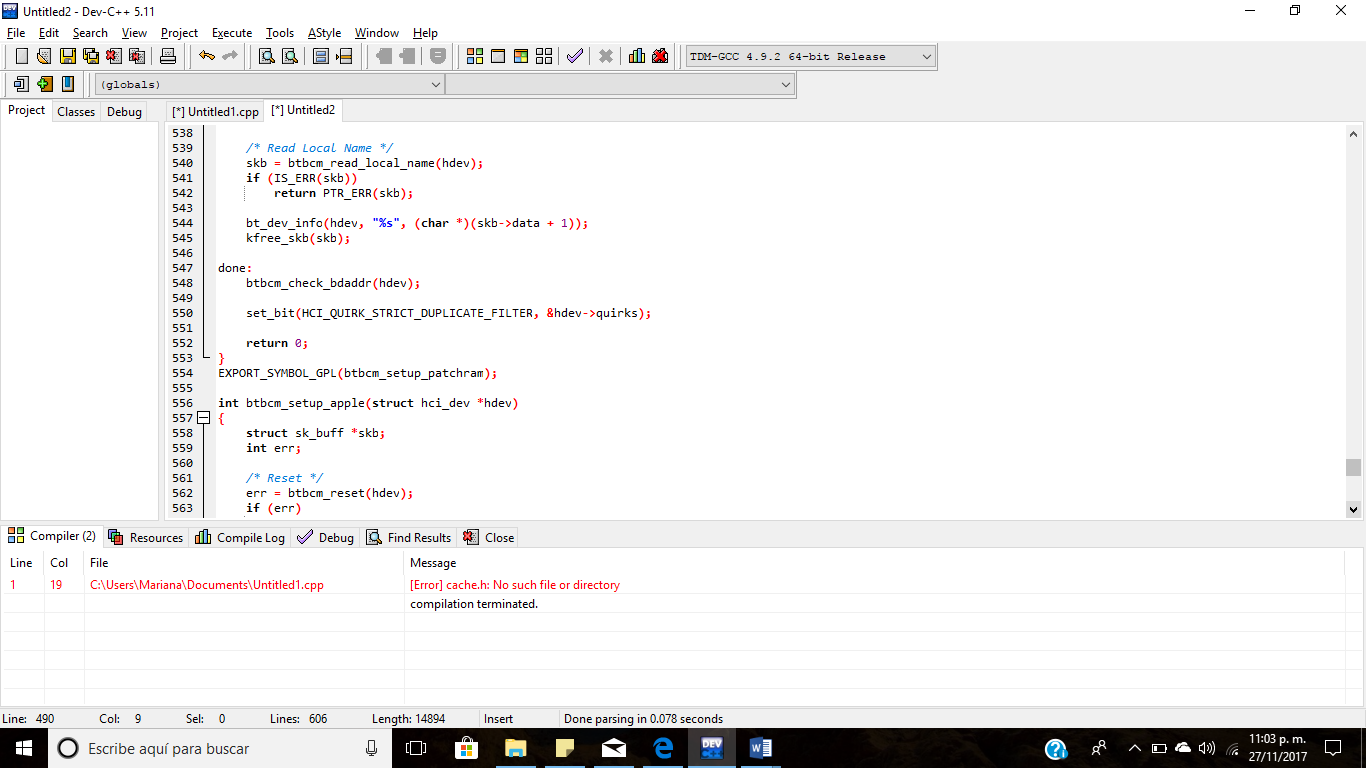


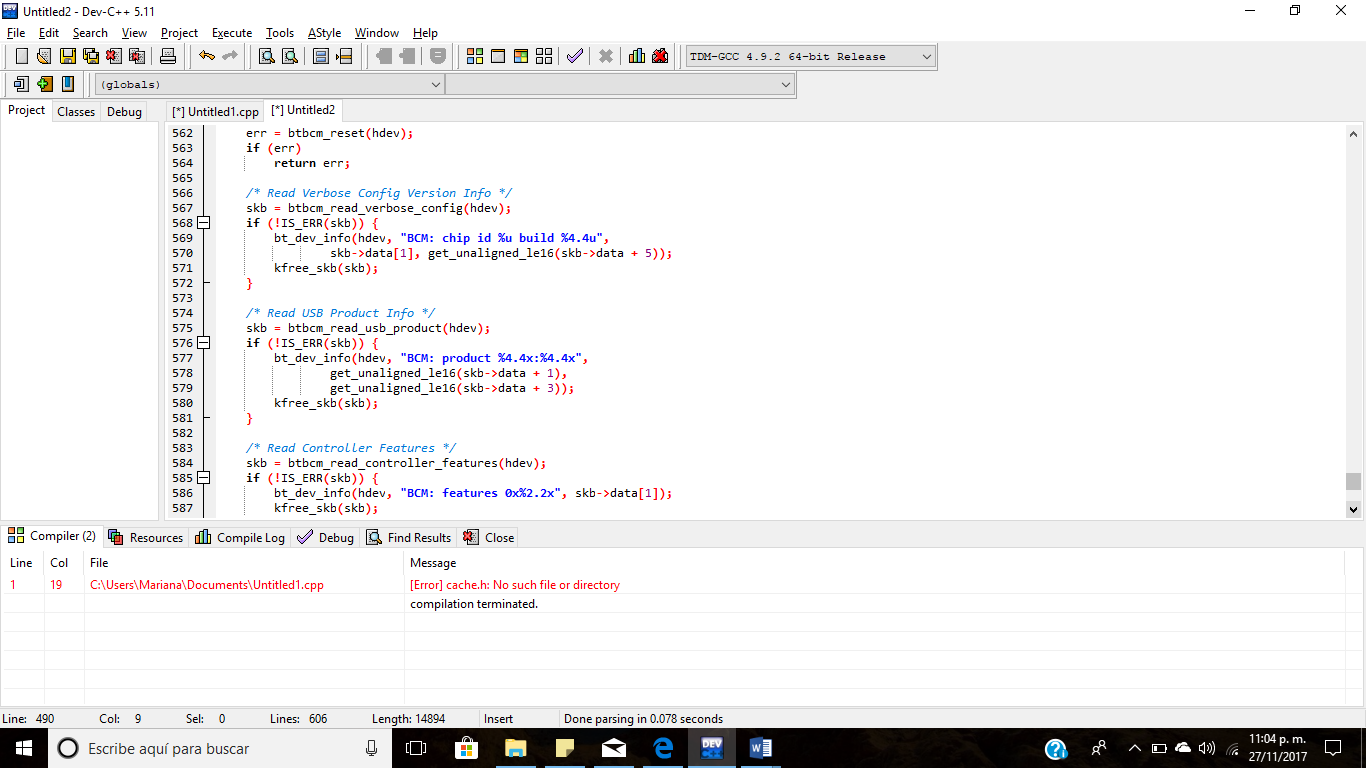


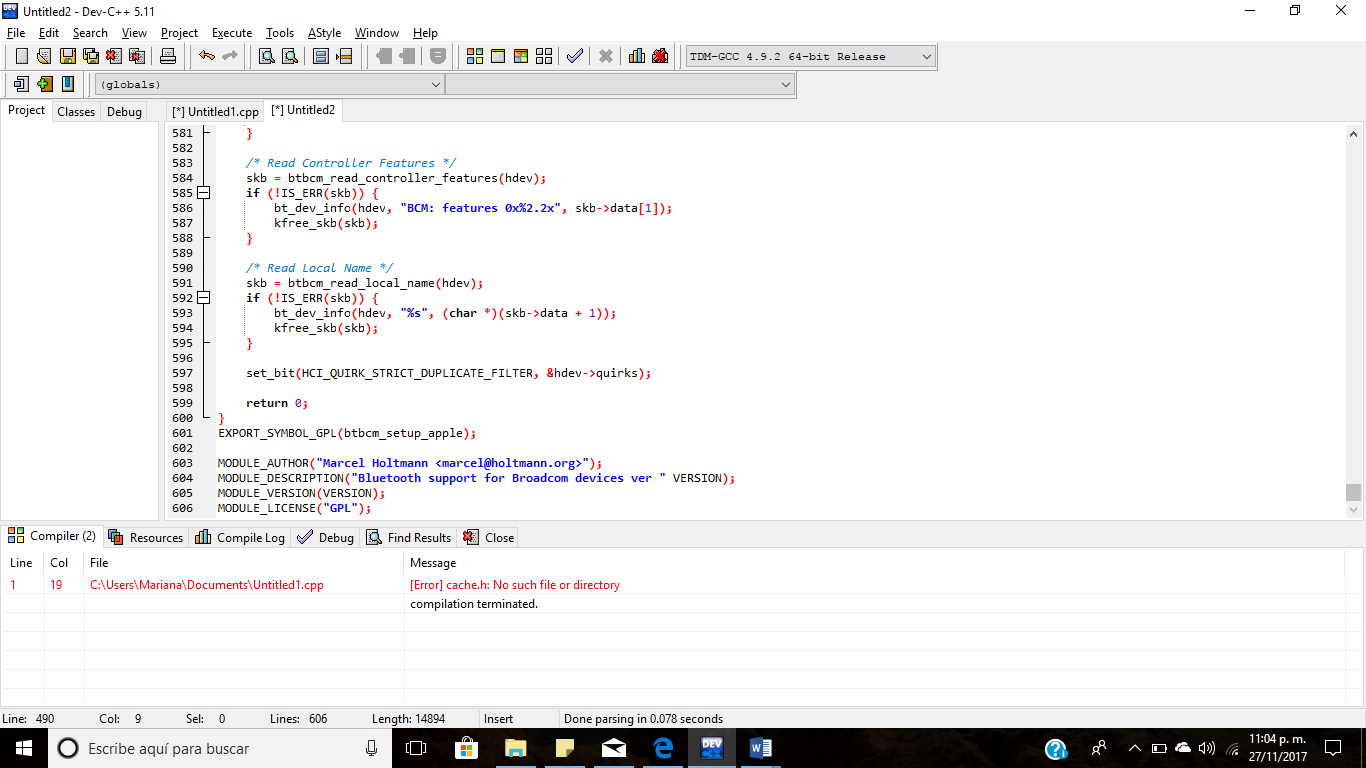












**“Todo programa es una función, pero no toda función es un programa”**

**¿Por qué?**

**Programa.**

Es aquel que intenta solucionar ó resolver todo el problema de “una”, de un solo “jalón”, ósea todo corrido.

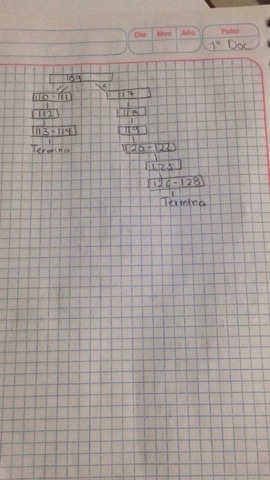
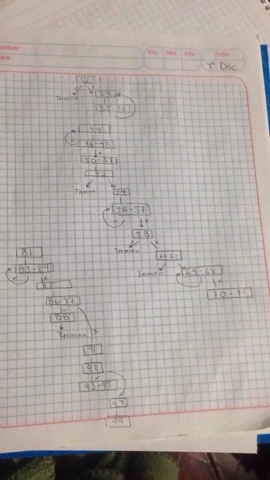
**Función.**

Es aquella que resuelve por partes el problema. También se podría decir que es una instrucción que resuelve tareas. La función, además, simplifica códigos.

Entonces se puede decir que la **diferencia** entre **programa** y **función** es que uno soluciona el problema por partes (función) y el otro lo resuelve en una sola vez.

**Archivo base85.c**

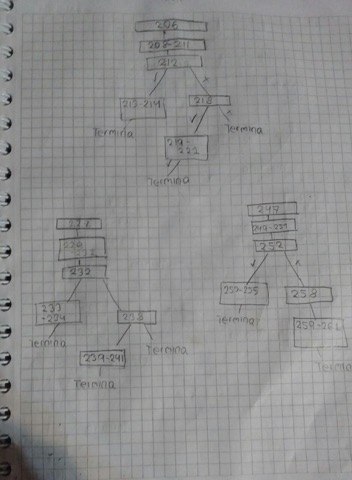
1. **Gráfica de control de flujo.**

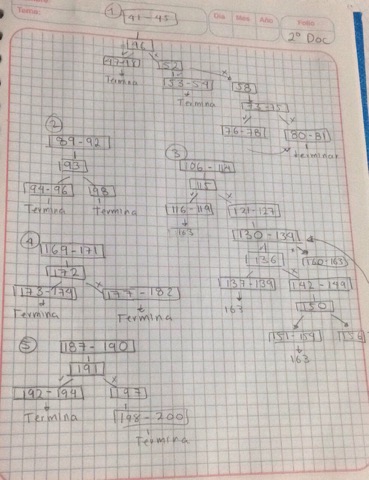
****

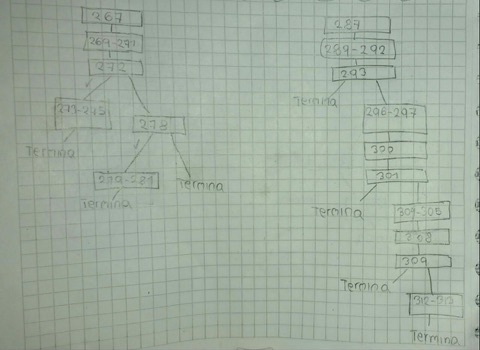
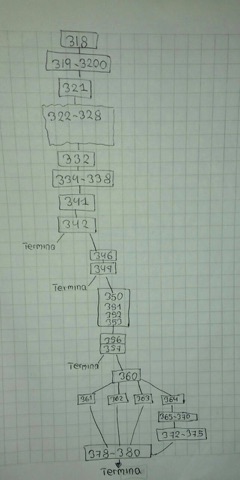
**2) Funciones que son llamadas en cada función.**

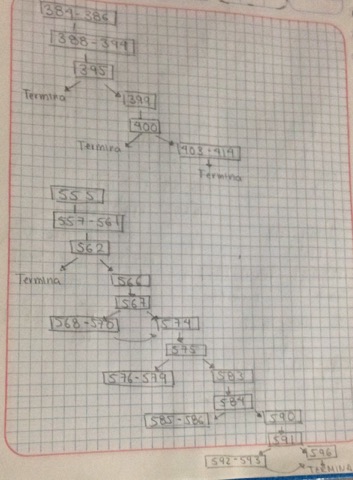
* static char de85[256];
* prep\_base85();
* say2("decode 85 <%.\*s>", len / 4 \* 5, buffer);
* say("\n");
* say("encode 85");
* char buf[1024];

**Archivo btbcm.c**

1. **Gráfica de control de flujo.**

****

****

****

**2) Funciones que son llamadas en cada función.**

* struct hci\_rp\_read\_bd\_addr \*bda;
* struct sk\_buff \*skb;
* skb = \_\_hci\_cmd\_sync(hdev, HCI\_OP\_READ\_BD\_ADDR, 0, NULL, HCI\_INIT\_TIMEOUT);
* bda = (struct hci\_rp\_read\_bd\_addr \*)skb->data;
* kfree\_skb(skb);
* EXPORT\_SYMBOL\_GPL(btbcm\_check\_bdaddr);
* kfree\_skb(skb);
* EXPORT\_SYMBOL\_GPL(btbcm\_set\_bdaddr);
* skb = \_\_hci\_cmd\_sync(hdev, 0xfc2e, 0, NULL, HCI\_INIT\_TIMEOUT);
* msleep(50);
* fw\_ptr = fw->data;
* fw\_size = fw->size;
* cmd = (struct hci\_command\_hdr \*)fw\_ptr;
* fw\_ptr += sizeof(\*cmd);
* fw\_size -= sizeof(\*cmd);
* cmd\_param = fw\_ptr;
* fw\_ptr += cmd->plen;
* fw\_size -= cmd->plen;
* return skb;
* opcode = le16\_to\_cpu(cmd->opcode);
* EXPORT\_SYMBOL(btbcm\_patchram);
* struct sk\_buff \*skb;
* const char \*name;
* skb = btbcm\_read\_local\_version(hdev);
* err = btbcm\_reset(hdev);
* ver = (struct hci\_rp\_read\_local\_version \*)skb->data;
* rev = le16\_to\_cpu(ver->hci\_rev);
* subver = le16\_to\_cpu(ver->lmp\_subver);
* err = btbcm\_read\_info(hdev);
* snprintf(fw\_name, len, "brcm/%s.hcd", hw\_name ? : "BCM");
* EXPORT\_SYMBOL\_GPL(btbcm\_initialize);
* const char \*name;
* const struct firmware \*fw;
* u16 subver, rev, pid, vid;
* const char \*hw\_name = NULL;
* struct sk\_buff \*skb;
* struct hci\_rp\_read\_local\_version \*ver;
* int i, err;
* err = btbcm\_reset(hdev);
* skb = btbcm\_read\_local\_version(hdev);
* skb = btbcm\_read\_local\_version(hdev);
* err = request\_firmware(&fw, fw\_name, &hdev->dev);
* btbcm\_patchram(hdev, fw);
* release\_firmware(fw);
* EXPORT\_SYMBOL\_GPL(btbcm\_setup\_patchram);