UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA

Ingeniería en Computación

Actividad 12 ANEXOS

Programación Estructurada

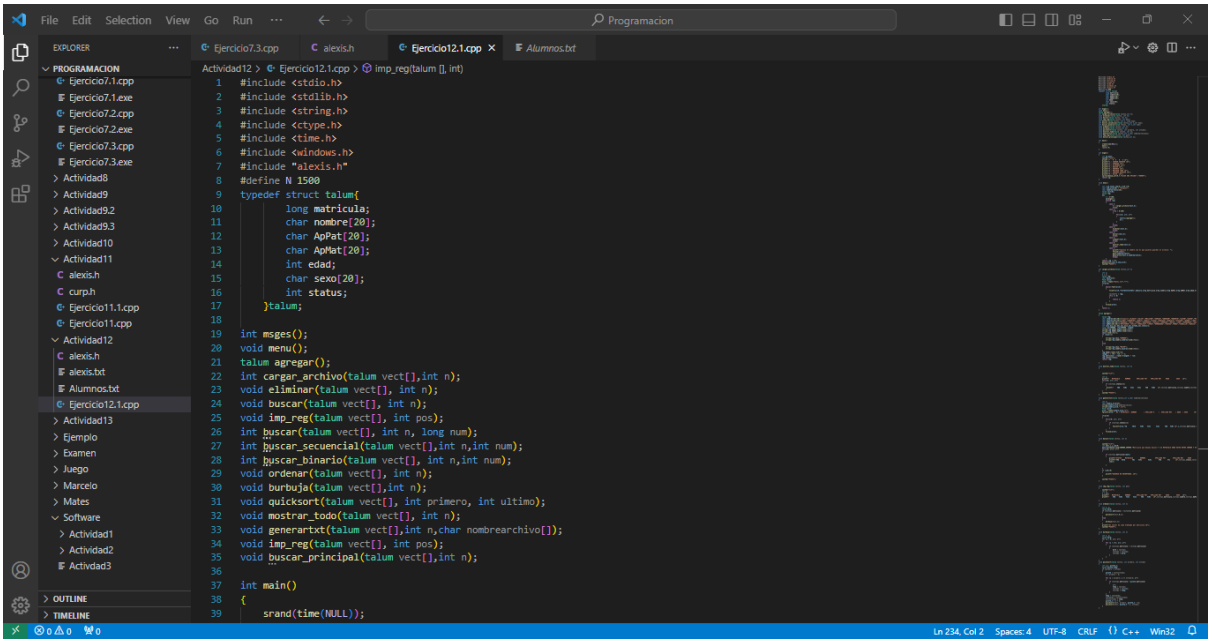
***

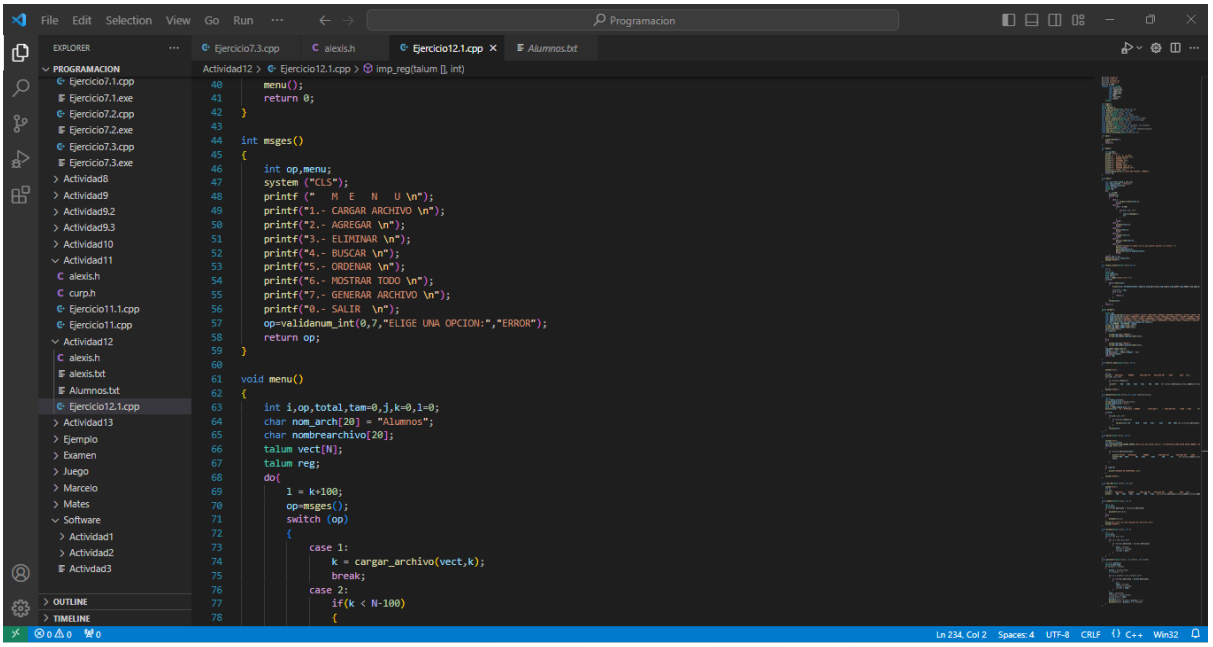
ALUMNO: Arredondo Urbalejo Isai Alexis

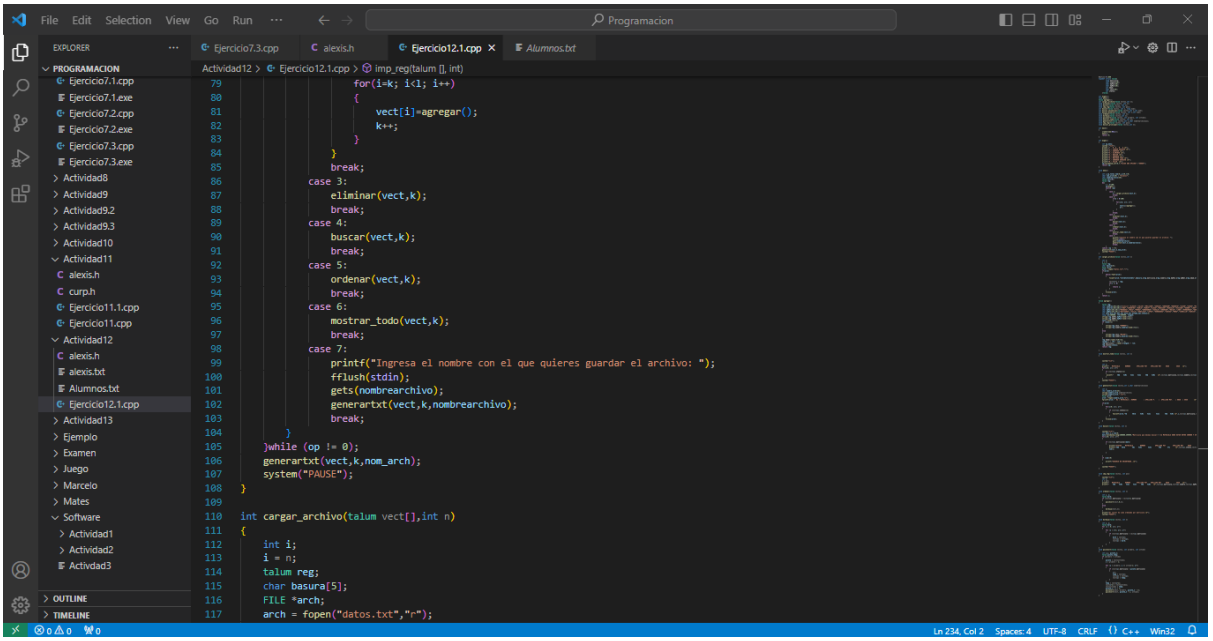
MATRÍCULA: 368747

GRUPO: 932

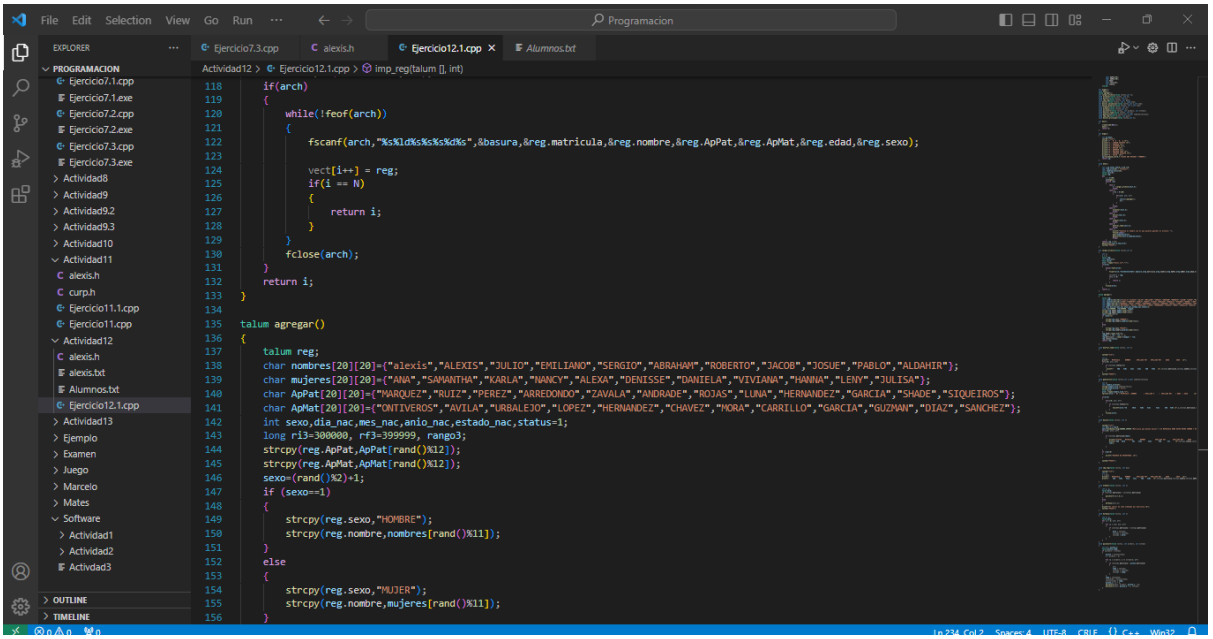
PROFESOR: Pedro Nunez Yepiz


File Edit Selection View Go Run
Programacion
EXPLORER
Ejercicio7.3.cpp
alexis.h
Ejercicio12.1.cpp
Alumnos.txt
Actividad12 > Ejercicio12.1.cpp > imp_reg(talum [], int)
PROGRAMACION
Ejercicio7.1.cpp
Ejercicio7.1.exe
Ejercicio7.2.cpp
Ejercicio7.2.exe
Ejercicio7.3.cpp
Ejercicio7.3.exe
Actividad8
Actividad9
Actividad9.2
Actividad9.3
Actividad10
Actividad11
alexis.h
curp.h
Ejercicio11.1.cpp
Ejercicio11.cpp
Actividad12
alexis.h
alexis.txt
Alumnos.txt
Ejercicio12.1.cpp
Actividad13
Ejemplo
Examen
Juego
Marcelo
Mates
Software
Actividad1
Actividad2
Actividad3
OUTLINE
TIMELINE
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <time.h>
#include <windows.h>
#include "alexis.h"
#define N 1500
typedef struct talum{
long matricula;
char nombre[20];
char ApPat[20];
char ApMat[20];
int edad;
char sexo[20];
int status;
}talum;
int msges();
void menu();
talum agregar();
int cargar_archivo(talum vect[],int n);
void eliminar(talum vect[], int n);
void buscar(talum vect[], int n);
void imp_reg(talum vect[], int pos);
int buscar(talum vect[], int n, long num);
int buscar_secuencial(talum vect[],int n,int num);
int buscar_binario(talum vect[], int n,int num);
void ordenar(talum vect[], int n);
void burbuja(talum vect[],int n);
void quicksort(talum vect[], int primero, int ultimo);
void mostrar_todo(talum vect[], int n);
void generartxt(talum vect[],int n,char nombrearchivo[]);
void imp_reg(talum vect[], int pos);
void buscar_principal(talum vect[],int n);
int main()
{
srand(time(NULL));
Ln 234, Col 2 Spaces: 4 UTF-8 CRLF C++ Win32

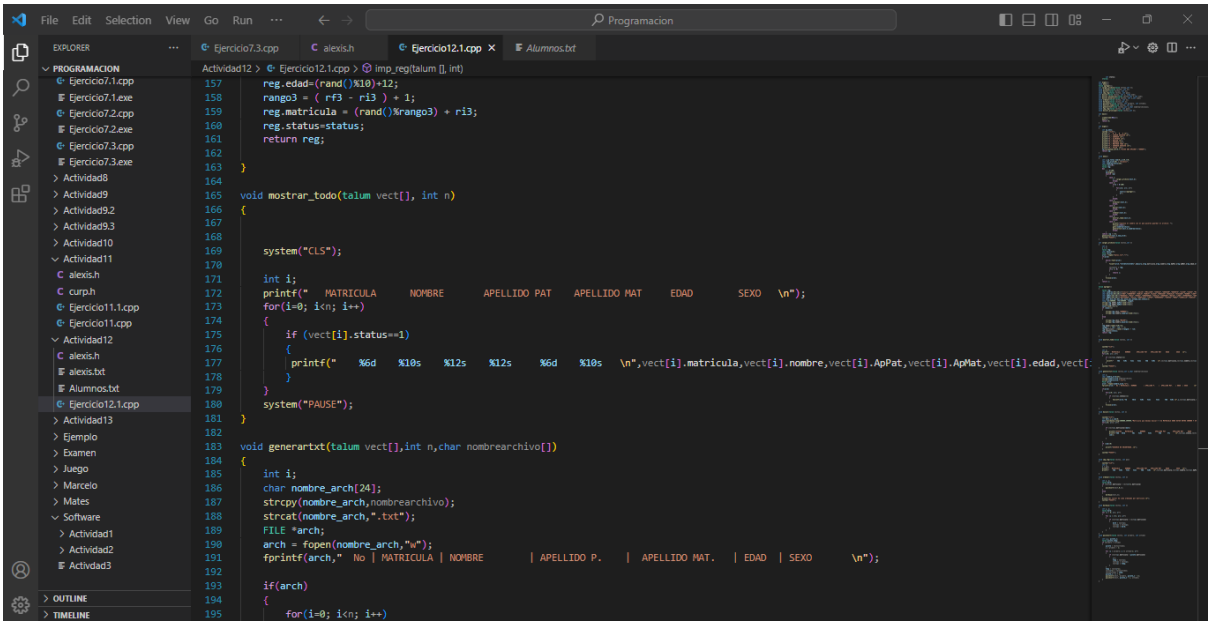

File Edit Selection View Go Run
Programacion
EXPLORER
Ejercicio7.3.cpp
alexis.h
Ejercicio12.1.cpp
Alumnos.txt
Actividad12 > Ejercicio12.1.cpp > imp_reg(talum [], int)
PROGRAMACION
Ejercicio7.1.cpp
Ejercicio7.1.exe
Ejercicio7.2.cpp
Ejercicio7.2.exe
Ejercicio7.3.cpp
Ejercicio7.3.exe
Actividad8
Actividad9
Actividad9.2
Actividad9.3
Actividad10
Actividad11
alexis.h
curp.h
Ejercicio11.1.cpp
Ejercicio11.cpp
Actividad12
alexis.h
alexis.txt
Alumnos.txt
Ejercicio12.1.cpp
Actividad13
Ejemplo
Examen
Juego
Marcelo
Mates
Software
Actividad1
Actividad2
Actividad3
OUTLINE
TIMELINE
menu();
return 0;
}
int msges()
{
int op,menu;
system ("CLS");
printf (" M E N U \n");
printf("1.- CARGAR ARCHIVO \n");
printf("2.- AGREGAR \n");
printf("3.- ELIMINAR \n");
printf("4.- BUSCAR \n");
printf("5.- ORDENAR \n");
printf("6.- MOSTRAR TODO \n");
printf("7.- GENERAR ARCHIVO \n");
printf("0.- SALIR \n");
op=validanum_int(0,7,"ELIGE UNA OPCION:","ERROR");
return op;
}
void menu()
{
int i,op,total,tam=0,j,k=0,l=0;
char nom_arch[20] = "Alumnos";
char nombrearchivo[20];
talum vect[N];
talum reg;
do{
l = k+100;
op=msges();
switch (op)
{
case 1:
k = cargar_archivo(vect,k);
break;
case 2:
if(k < N-100)
{
Ln 234, Col 2 Spaces: 4 UTF-8 CRLF C++ Win32



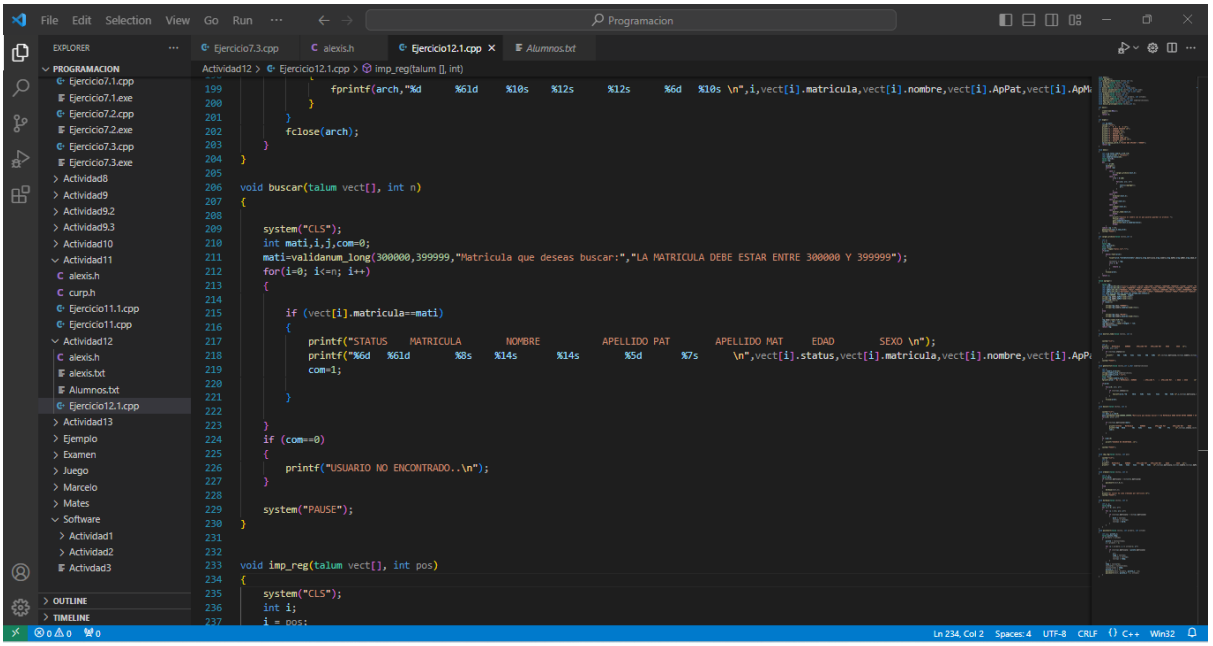
```
                    for(i=k; i<l; i++)
                    {
                        vect[i]=agregar();
                        k++;
                    }
                }
                break;
            case 3:
                eliminar(vect,k);
                break;
            case 4:
                buscar(vect,k);
                break;
            case 5:
                ordenar(vect,k);
                break;
            case 6:
                mostrar_todo(vect,k);
                break;
            case 7:
                printf("Ingresa el nombre con el que quieres guardar el archivo: ");
                fflush(stdin);
                gets(nombrearchivo);
                generartxt(vect,k,nombrearchivo);
                break;
        }
    }while (op != 0);
    generartxt(vect,k,nom_arch);
    system("PAUSE");
}

int cargar_archivo(talum vect[],int n)
{
    int i;
    i = n;
    talum reg;
    char basura[5];
    FILE *arch;
    arch = fopen("datos.txt","r");
```



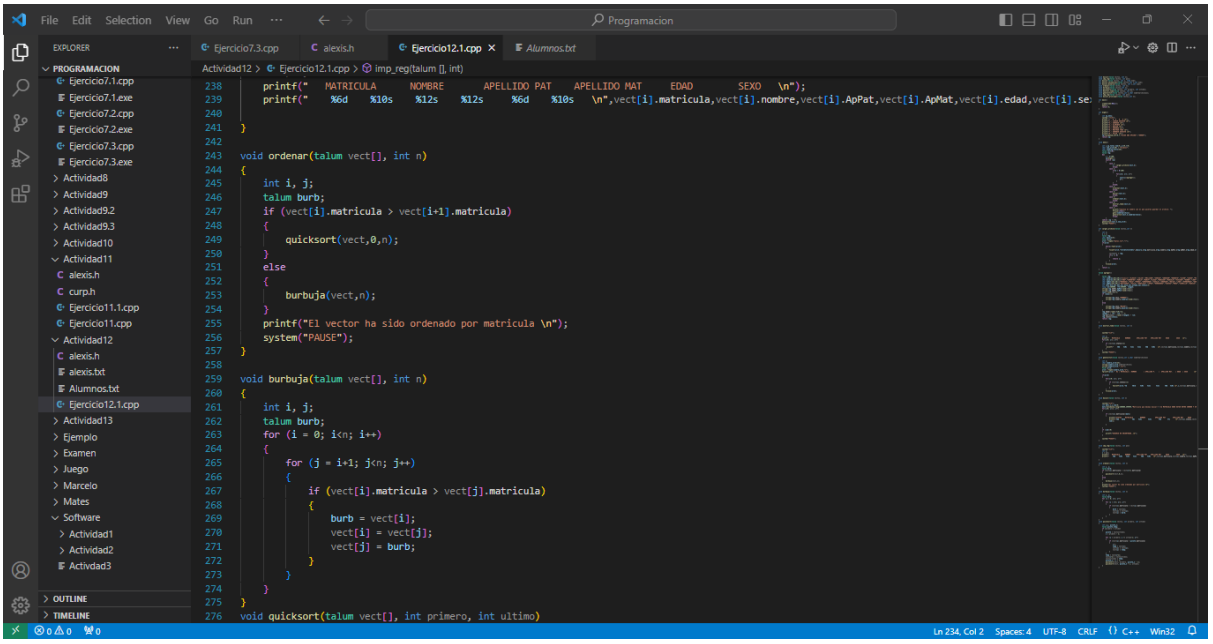
```
    if(arch)
    {
        while(!feof(arch))
        {
            fscanf(arch,"%s%ld%s%s%s%d%s",&basura,&reg.matricula,&reg.nombre,&reg.ApPat,&reg.ApMat,&reg.edad,&reg.sexo);

            vect[i++] = reg;
            if(i == N)
            {
                return i;
            }
        }
        fclose(arch);
    }
    return i;
}

talum agregar()
{
    talum reg;
    char nombres[20][20]={"alexis","ALEXIS","JULIO","EMILIANO","SERGIO","ABRAHAM","ROBERTO","JACOB","JOSUE","PABLO","ALDAHIR"};
    char mujeres[20][20]={"ANA","SAMANTHA","KARLA","NANCY","ALEXA","DENISSE","DANIELA","VIVIANA","HANNA","LENY","JULISA"};
    char ApPat[20][20]={"MARQUEZ","RUIZ","PEREZ","ARREDONDO","ZAVALA","ANDRADE","ROJAS","LUNA","HERNANDEZ","GARCIA","SHADE","SIQUEIROS"};
    char ApMat[20][20]={"ONTIVEROS","AVILA","URBALEJO","LOPEZ","HERNANDEZ","CHAVEZ","MORA","CARRILLO","GARCIA","GUZMAN","DIAZ","SANCHEZ"};
    int sexo,dia_nac,mes_nac,anio_nac,estado_nac,status=1;
    long ri3=300000, rf3=399999, rango3;
    strcpy(reg.ApPat,ApPat[rand()%12]);
    strcpy(reg.ApMat,ApMat[rand()%12]);
    sexo=(rand()%2)+1;
    if (sexo==1)
    {
        strcpy(reg.sexo,"HOMBRE");
        strcpy(reg.nombre,nombres[rand()%11]);
    }
    else
    {
        strcpy(reg.sexo,"MUJER");
        strcpy(reg.nombre,mujeres[rand()%11]);
    }
```



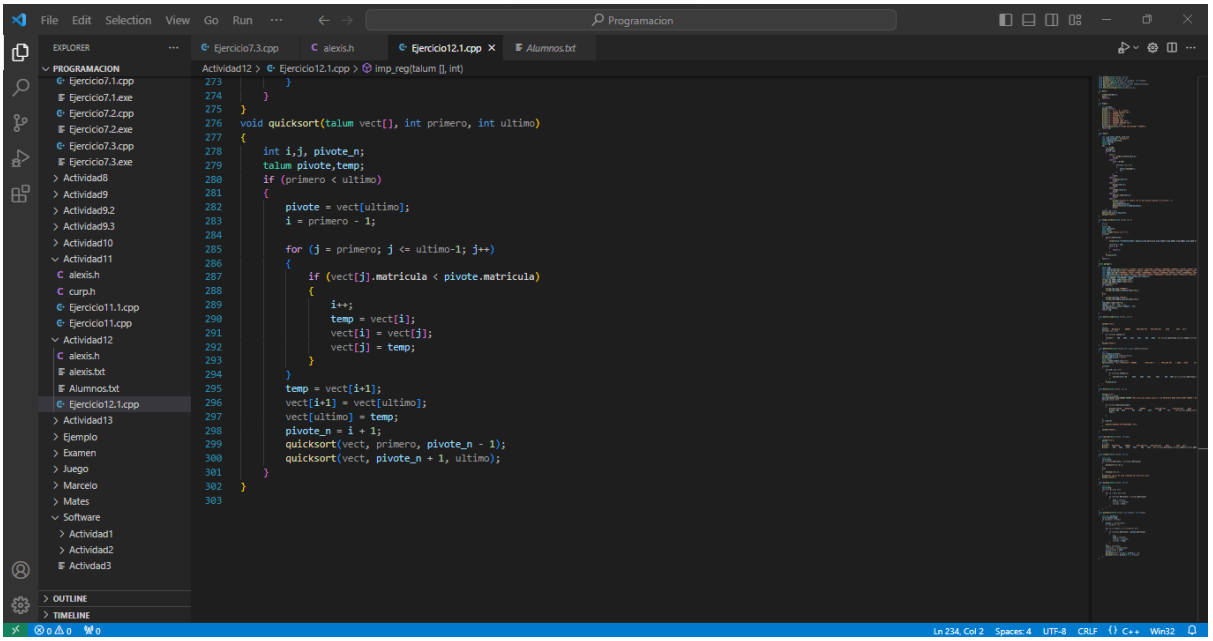
```
    reg.edad=(rand()%10)+12;
    rango3 = ( rf3 - ri3 ) + 1;
    reg.matricula = (rand()%rango3) + ri3;
    reg.status=status;
    return reg;

}

void mostrar_todo(talum vect[], int n)
{


    system("CLS");

    int i;
    printf("   MATRICULA      NOMBRE       APELLIDO PAT    APELLIDO MAT     EDAD        SEXO   \n");
    for(i=0; i<n; i++)
    {
        if (vect[i].status==1)
        {
         printf("    %6d    %10s    %12s    %12s     %6d    %10s   \n",vect[i].matricula,vect[i].nombre,vect[i].ApPat,vect[i].ApMat,vect[i].edad,vect[
        }
    }
    system("PAUSE");
}

void generartxt(talum vect[],int n,char nombrearchivo[])
{
    int i;
    char nombre_arch[24];
    strcpy(nombre_arch,nombrearchivo);
    strcat(nombre_arch,".txt");
    FILE *arch;
    arch = fopen(nombre_arch,"w");
    fprintf(arch,"  No | MATRICULA | NOMBRE        | APELLIDO P.    | APELLIDO MAT.   | EDAD  | SEXO       \n");

    if(arch)
    {
        for(i=0; i<n; i++)
```



```
                fprintf(arch,"%d      %6ld     %10s    %12s     %12s      %6d    %10s \n",i,vect[i].matricula,vect[i].nombre,vect[i].ApPat,vect[i].ApM
            }
        }
        fclose(arch);
    }
}

void buscar(talum vect[], int n)
{

    system("CLS");
    int mati,i,j,com=0;
    mati=validanum_long(300000,399999,"Matricula que deseas buscar:","LA MATRICULA DEBE ESTAR ENTRE 300000 Y 399999");
    for(i=0; i<=n; i++)
    {

        if (vect[i].matricula==mati)
        {
            printf("STATUS    MATRICULA        NOMBRE           APELLIDO PAT        APELLIDO MAT     EDAD        SEXO \n");
            printf("%6d    %6ld        %8s    %14s       %14s        %5d        %7s       \n",vect[i].status,vect[i].matricula,vect[i].nombre,vect[i].ApP
            com=1;

        }

    }
    if (com==0)
    {
        printf("USUARIO NO ENCONTRADO..\n");
    }

    system("PAUSE");
}


void imp_reg(talum vect[], int pos)
{
    system("CLS");
    int i;
    i = pos;
```



```
    printf("   MATRICULA      NOMBRE       APELLIDO PAT     APELLIDO MAT     EDAD        SEXO   \n");
    printf("    %6d    %10s    %12s    %12s    %6d    %10s    \n",vect[i].matricula,vect[i].nombre,vect[i].ApPat,vect[i].ApMat,vect[i].edad,vect[i].se

}

void ordenar(talum vect[], int n)
{
    int i, j;
    talum burb;
    if (vect[i].matricula > vect[i+1].matricula)
    {
        quicksort(vect,0,n);
    }
    else
    {
        burbuja(vect,n);
    }
    printf("El vector ha sido ordenado por matricula \n");
    system("PAUSE");
}

void burbuja(talum vect[], int n)
{
    int i, j;
    talum burb;
    for (i = 0; i<n; i++)
    {
        for (j = i+1; j<n; j++)
        {
            if (vect[i].matricula > vect[j].matricula)
            {
                burb = vect[i];
                vect[i] = vect[j];
                vect[j] = burb;
            }
        }
    }
}
void quicksort(talum vect[], int primero, int ultimo)
```



```
        }
    }
}
void quicksort(talum vect[], int primero, int ultimo)
{
    int i,j, pivote_n;
    talum pivote,temp;
    if (primero < ultimo)
    {
        pivote = vect[ultimo];
        i = primero - 1;

        for (j = primero; j <= ultimo-1; j++)
        {
            if (vect[j].matricula < pivote.matricula)
            {
                i++;
                temp = vect[i];
                vect[i] = vect[j];
                vect[j] = temp;
            }
        }
        temp = vect[i+1];
        vect[i+1] = vect[ultimo];
        vect[ultimo] = temp;
        pivote_n = i + 1;
        quicksort(vect, primero, pivote_n - 1);
        quicksort(vect, pivote_n + 1, ultimo);
    }
}
```