/\*\*Este Proyecto Estudia el Concepto de las Uniones (UNIONS)\*\*/

///Union Definition

// 1. User-defined Datatype

// 2. A collection of variables stored in the same region of memory

#include <stdio.h>

#include <stdlib.h>

int main()

{

    /\*differencesBetweenStructsAndUnions();\*/

    moreUsesForUnions();

    return 0;

}

/\*\*Este Procedimiento Aborda las Diferencias entre Estructuras y Uniones\*\*/

struct point\_struct{ /// In a struct, each element has a separate space in memory

    int x;           /// Thus, there are 8 bytes in memory for both integers (4 bytes each)

    int y;

};

union point\_union{ /// In a union, each element points toward the same region in memory

    int x;         /// Thus, there are only 4 bytes in memory for both integers (4 bytes each)

    int y;         /// Due to this, when modifying one element, every other element of a union gets modified

};

void differencesBetweenStructsAndUnions()

{

    struct point\_struct point1;

    union point\_union point2;

    point1.x = 5;

    point1.y = 10;

    printf("point\_struct: (%i,%i) \n", point1.x, point1.y);

    point2.y = 9;

    printf("point\_union: (%i,%i) \n", point2.x, point2.y);

    point1.y = 15;

    printf("Despues de modificar --> point\_struct: (%i,%i) \n", point1.x, point1.y);

    point2.x = 1;

    printf("Despues de modificar --> point\_union: (%i,%i)", point2.x, point2.y);

    printf("\n \n");

}

/\*\*Este Procedimiento Muestra Otros Usos de las Uniones\*\*/

#include<stdio.h>

typedef union cPerson{ ///Uso de TYPEDEF con UNIONS

    int ID;

    char cName[20];

}person;

void moreUsesForUnions()

{

    int f;

    person person1;

    person \*pointerToPerson1; ///Podemos usar punteros que guarden la direccion de memoria de una UNION

    printf("Ingresa el ID: ");

    scanf("%i", &person1.ID);

    fflush(stdin);

    printf("ID ingresado: %i \n", person1.ID);

    pointerToPerson1 = &person1;

    (\*pointerToPerson1).ID = 11; /// Esta es UNA  forma de modificar la informacion de la UNION a la que apunta un puntero

    pointerToPerson1->ID = 12;   /// Esta es OTRA forma de modificar la informacion de la UNION a la que apunta un puntero

    printf("Despues de modificar via punteros --> ID ingresado: %i \n", person1.ID);

    printf("\n \n");

    person personArray[5];

    for (f = 0; f < 5; f++)

    {

        printf("Ingresa el ID #%i: ", f + 1);

        scanf("%i", &personArray[f].ID);

        fflush(stdin);

    }

    printf("Impresion de IDs. \n");

    for (f = 0; f < 5; f++)

        printf("\tID #%i: %i\n", f + 1, personArray[f].ID);

    printf("\n \n");

}

/\*\*Este Procedimiento Muestra como se Define una Union\*\*/

#include<stdio.h>

//Defining the Template

union info{

    char cName[20];

    int age;

};

void definingUnions()

{

    //Defining a Variable of Type UNION INFO

    union info person1;

    printf("\n \n");

}