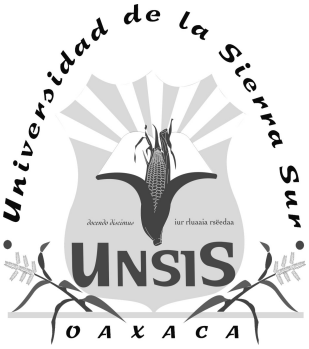

UNIVERSIDAD DE LA SIERRA SUR

Instituto de Informática

Programa de conversión de bases

Labortorio de Sistemas Digitales

Alumnos:
Elietzer Jared, Elio Justino

Profesor: Dr. Alejandro Jarillo Silva

Grupo: 306

17 de octubre de 2022

Índice

Alumno:

Elietzer Jared

Elio Justino

Grupo 306

Convertidor de bases

1. Introducción

Se desarrollará un programa de conversión de bases, el cual admitirá decimal, octal, binario, hexadecimal y formato BCD. Para poder desplegarlo con formato se utilizará la tecnología GTK+, biblioteca de el lenguaje de programación C.

2. Objetivos

1. Realizar una calculadora capaz de cambiar la base de un dado entre las bases: Decimal, Octal, Hexadecimal, y Formato BCD.

2. Otorgar un diseño simple y útil a la interfaz gráfica para facilitar el manejo del programa.

3. Obtener un algoritmo eficiente capaz de abstraer los conceptos teóricos y aplicarlos de forma correcta.

3. Desarrollo

3.1. Planteamiento del problema

Desarrollar un programa capaz de convertir un número de base decimal, Octal, Binaria, Hexadecimal o formato BCD al resto de las bases, esto a través de una interfaz gráfica. La implementación se realizará a través de C y su biblioteca gráfica GTK+ en su versión 3.24.20.

3.2. Diseño y creación de algoritmos

Al inicio se pensaba darle un diseño de calculadora al programa, pero debido a que tendrá bases específicas, se optó por un diseño en el cual él mismo programa determinará la base de entrada, para posteriormente calcular las demás bases.

El programa tendrá un diseño simple, que permita al usuario ingresar de cualquier base admitida para obtenerlo en todas las demás bases disponibles. Para lograr dicha flexibilidad se optó por el diseño mostrado en la figura 1

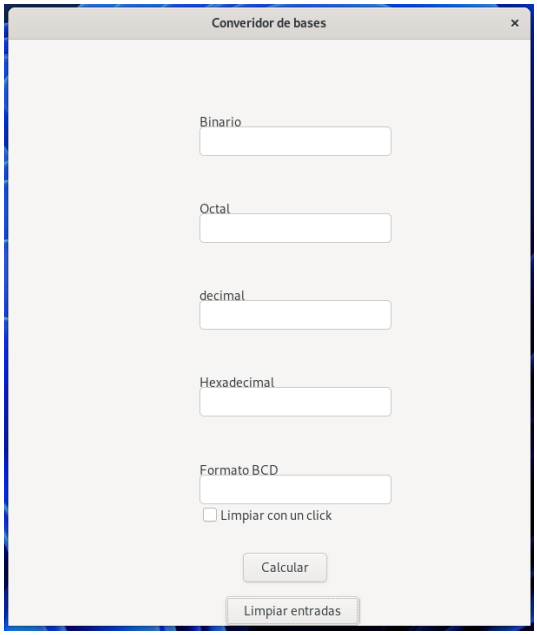

Figura 1: Diseño del programa

Para limpiar los campos de entrada de tex-

to se crearon dos formas, una mediante un botón, y otra a través de un clic sobre cualquiera de los campos. El cambio de método se da a través del check con la leyenda: Limpiar con un clic.

Para el proceso de conversión se existen muchas formas de realizarlo, sin embargo se observó que pasar de Decimal a Binario, Octal y Hexadecimal se debe dividir el número decimal entre la base solicitada e ir concatenando los módulos de cada división hasta obtener cero, como se muestra en la figura 2, donde se convierte de decimal a octal.

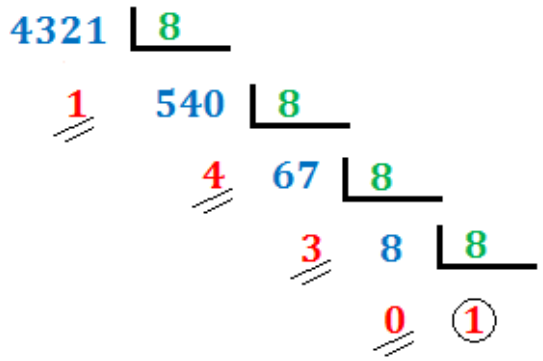

Figura 2: Decimal a Octal

```c
void DecToHexaOctaBin(gint64 decimal, int base){
    GString *mod = g_string_new("");
    int result;
    result = decimal;
    char aux[10];
    while (result != 0) {
        // Conversión y obtención de modulo
        sprintf(aux, "%d", (result % base));
        result = result / base;
        g_print("mod: %s\n", aux);

        // Validación para base Hexadecimal
        if (strcmp(aux, "10") == 0) {
            strcpy(aux, "A");
        } else if (strcmp(aux, "11") == 0) {
            strcpy(aux, "B");
        } else if (strcmp(aux, "12") == 0) {
            strcpy(aux, "C");
        } else if (strcmp(aux, "13") == 0) {
            strcpy(aux, "D");
        } else if (strcmp(aux, "14") == 0) {
            strcpy(aux, "E");
        } else if (strcmp(aux, "15") == 0) {
            strcpy(aux, "F");
        } else {
            g_print("normal ");
        }
        g_print("mod2: %s\n", aux);

        mod = g_string_prepend(mod, aux);
    }
    // Asignación del valor a la salida correspondiente
    if (base == 16) {
        gtk_entry_set_text(GTK_ENTRY(txtHexa), mod->str);
    } else if (base == 8) {
        gtk_entry_set_text(GTK_ENTRY(txtOcta), mod->str);
    } else if (base == 2){
        gtk_entry_set_text(GTK_ENTRY(txtBin), mod->str);
    }

```

Figura 3: Algoritmo de conversión de bases

Con dicha forma de convertir las bases desde un número decimal se llegó a un algoritmo el cual convierte desde decimal a octal, binario y Hexadecimal, además de mostrar el resultado al usuario y de forma opcional regresarlo mediante un return en caso de ser necesario, el cual se muestra en la figura 3.

Con el uso de este algoritmo solamente faltaría pasar de base Octal, Binario y Hexadecimal a binario, una vez echo esto el programa estaría casi completo, por lo que el primero en desarrollarse será de Binario a Decimal. Lo primero que se realizó fue buscar el punto decimal en la cadena binaria para futuras versiones. En este caso solo nos centraremos en los números enteros. Una vez que tenemos

una cadena binaria entera se recorre de derecha a izquierda para ir elevando la base a su potencia correspondiente. Figura 4

```
// Create needed objects and vars
    GString *afterPoint = g_string_new(value -> str);
    GString *beforePoint = g_string_new(value -> str);

    gboolean flag = FALSE;
    gfloat help = 0;
    gint pot = 0;
    gint result = 0;
    // Separete the string from point
    for (gint i = 0; i < value->len; i++) {
        if (value -> str[i] == '.'){
            beforePoint = g_string_truncate(beforePoint, i);
            afterPoint = g_string_erase(afterPoint, 0, i + 1);
            flag = TRUE;
            break;
        }
    }
    // Convert the part before at point
    for (gint i = (beforePoint -> len) - 1; i >= 0; i--) {
        if (beforePoint -> str[i] == '1') {
            pot = (beforePoint->len - 1) - i;
            help = pow(2, pot);
            result += help;
        }
    }
```

Figura 4: Algoritmo de binario a decimal

El resto de los algoritmos que convierten a binario se realizaron de forma similar. Una vez que fue posible convertir a decimal se utilizó el algoritmo de la figura 3 para llegar a las demás bases.

Para el formato BCD primero tenemos que tomar en cuenta que la cadena binaria se divide en grupos de 4 elementos para posteriormente transformarlos a decimal de forma separada y concatenar dichos números para obtener el resultado.

Se desarrolló un algoritmo que asegura que la cadena contendrá grupos de 4 elementos, posteriormente utiliza el mismo algoritmo de conversión de Binario a Decimal (figura 4) por grupos de 4 en 4 elementos. Por último concatena los resultados separados y muestra el resultado. Algoritmo en la figura 5. De forma adicional se transformó el formato bcd a Octal, Binario y Hexadecimal utilizando la función de la figura 3 y el valor decimal obtenido.

```
int bcdToDecimal(GString *value, int flag) {
    GString *result =  g_string_new("");
    GString *trunc =  g_string_new("");
    GString *request = g_string_new("");;
    // Valid multipl to 4
    gint difference = 0;
    for (gint i = 4; i <= (value->len + 4); i += 4) {
        if (i >= (value->len - 1)) {
            difference = i - (value->len);
        }
    }
    // Add char's for complete multi the 3
    if (difference < 4 && difference != 0) {
        for (gint i = 1; i <= difference; i++) {
            g_string_prepend_c(value, '0');
        }
    }
    for (gint i = 4; i < (value -> len) + 4; i+=4) {
        trunc = g_string_insert(trunc, 0, value->str);
        // Recorta 4 numeros
        trunc = g_string_erase(trunc, i, 0 - 1);
        trunc = g_string_erase(trunc, 0, i - 4);
        request = g_string_append_c(request, binToDecimal(trunc, 1));
        // Vacia la cadena
        trunc = g_string_erase(trunc, 0, 0 - 1);
    }
    g_print("Valor: %s\n", request->str);
    if (flag == 0) {
        gtk_entry_set_text(GTK_ENTRY(txtDec), request->str);
    } else {
        int retu = 0;
        retu = atoi(request->str);
        g_print ("return: %d\n", retu);
        return retu;
    }
}
```

Figura 5: Algoritmo de BCD a decimal

Para convertir de los demás formatos a BCD se utiliza un algoritmo que los convierte a base decimal, donde para pasar de decimal a BCD se realizó un algoritmo que convierte decimal a formato BCD, como se muestra en la figura 6. Es posible reutilizar dicho algoritmo para convertir de las demáss bases a formato BCD pasando por la base decimal y así obtener su equivalente.

```c
void decToBcd (int val) {
    char str[20];
    sprintf(str, "%d", val);
    GString *value = g_string_new(str);
    GString *request = g_string_new("");
    int arg = 0;
    GString *aux= g_string_new("");

    for (int i = 0; i < (value->len); i++) {
        // Cast gchar to int
        arg = value->str[i] - '0';
        if (arg == 0) {
            aux = g_string_append(aux, "0000");
        } else {
            aux = g_string_append(aux, DecToHexaOctaDirect(arg, 2, 1));
            g_print("pre aux: %s\n", aux->str);
            if((aux->len - 1) < 4){
                for (int j = (aux->len); j < 4; j++) {
                    aux = g_string_prepend_c(aux, '0');
                }
            }
        }
        request = g_string_append(request, aux->str);
        g_print("arg: %d request: %s aux: %s\n", arg, request->str, aux->str);
        aux = g_string_erase(aux, 0, -1);
    }
    gtk_entry_set_text(GTK_ENTRY(txtBcd), request->str);
}
```

Figura 6: Algoritmo de decimal a BCD

4. Resultados

El programa actúa según lo esperado, siendo capaz de convertir de cualquier base al resto (dentro de las admitidas. Figura 7). Aún es posible mejorar el código utilizando otras herramientas que nos provee C, además de agregar aperaciones al programa como lo son la suma, resta, etc. Pero los objetivos para este proyecto se cumplieron. El código fuente se encuentra al final del archivo donde se estarán publicando las próximas versiones.

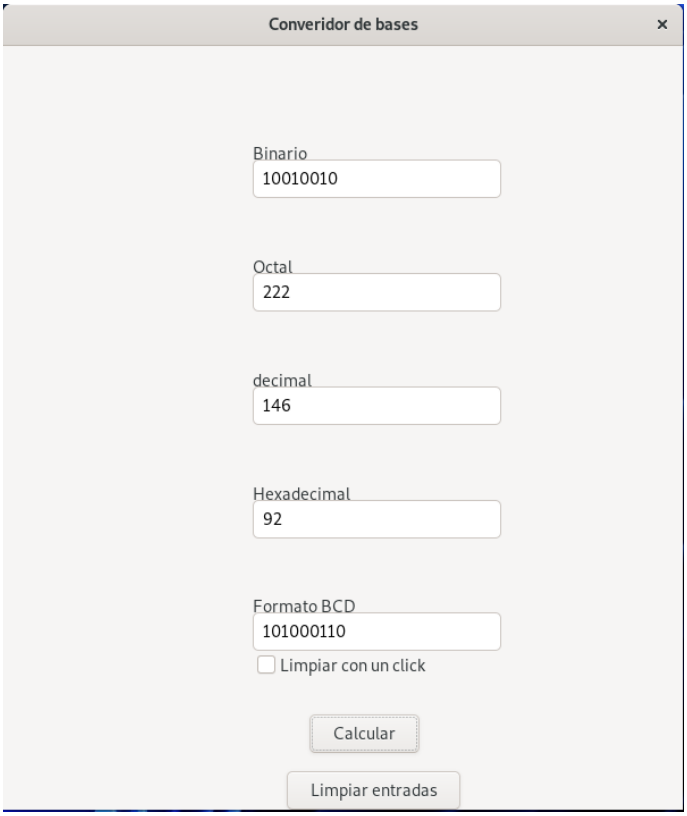

Figura 7: Muestra de función del programa finalizado

5. Conclusiones

El programa se completó con éxito, si bien no se realizaron conversiones entre algunas bases de forma directa, el objetivo del programa es poder realizar el cambio de base de un número de la manera más eficiente posible, que es lo que se realizó en dicho algoritmo, reciclar y generalizar un algoritmo para mejorar su legibilidad y eficiencia.

Código fuente: https://github.com/UnsisWorks/convert_base_gtk