



UNIVERSIDAD DE LA SIERRA SUR

Instituto de Informática

Programa de conversión de bases

Labortorio de Sistemas Digitales

Alumnos:

Elietzer Jared, kevin Emmanuel

Profesor: Dr. Alejandro Jarillo Silva

Grupo: 306

17 de octubre de 2022

Índice

1. Introducción	2
2. Objetivos	2
3. Desarrollo	2
3.1. Planteamiento del problema	2
3.2. Diseño y creación de algoritmos	2
4. Resultados	7
5. Conclusiones	8

Alumno:

Elietzer Jared

Kevin Emmanuel

Grupo 306

Convertidor de bases

1. Introducción

Se desarrollara un programa de conversión 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, Hexadecim, y Formato BCD
2. Otorgar un diseño simple y util a la interfaz grafica para la facilitar el manejo del programa

3. Desarrollo

3.1. Planteamiento del problema

Desarrollar un programa capaz de convertir un número de base decimal, Octal, Binaria, Hexadecimal o formato BCD a el 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 a el programa, pero debido que tendrá bases específicas se opto por un diseño en el cual el mismo programa determinara la base de entrada, para posteriormente calcular las demás bases.

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

Figura 1: Diseño del programa

Para limpiar los campos de entrada de texto se crearon dos formas, una mediante un botón, y otra a través de un click sobre cualquiera de los campos. El cambio de metodo se da a través de el check con la leyenda: Limpiar con un click.

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

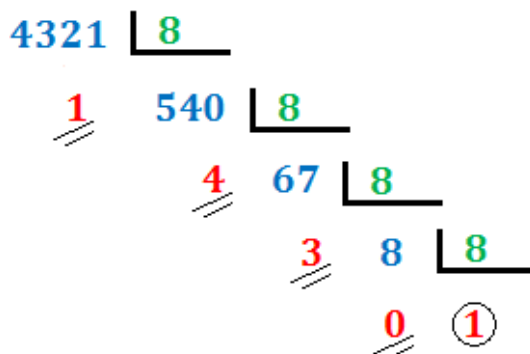


Figura 2: Decimal a Octal

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

```
1 void DecToHexaOctaBin(gint64 decimal, int base){
2     GString *mod = g_string_new("");
3     int result;
4     result = decimal;
5     char aux[10];
6     while (result != 0) {
7         // Conversión y obtención de modulo
8         sprintf(aux, "%d", (result % base));
9         result = result / base;
10        g_print("mod: %s\n", aux);
11
12        // Validación para base Hexadecimal
13        if (strcmp(aux, "10") == 0) {
14            strcpy(aux, "A");
15        } else if (strcmp(aux, "11") == 0) {
16            strcpy(aux, "B");
17        } else if (strcmp(aux, "12") == 0) {
18            strcpy(aux, "C");
19        } else if (strcmp(aux, "13") == 0) {
20            strcpy(aux, "D");
21        } else if (strcmp(aux, "14") == 0) {
22            strcpy(aux, "E");
23        } else if (strcmp(aux, "15") == 0) {
24            strcpy(aux, "F");
25        } else {
26            g_print("normal ");
27        }
28        g_print("mod2: %s\n", aux);
29
30        mod = g_string_prepend(mod, aux);
31    }
32    // Asignación del valor a la salida correspondiente
33    if (base == 16) {
34        gtk_entry_set_text(GTK_ENTRY(txtHexa), mod->str);
35    } else if (base == 8) {
36        gtk_entry_set_text(GTK_ENTRY(txtOcta), mod->str);
37    } else if (base == 2){
38        gtk_entry_set_text(GTK_ENTRY(txtBin), mod->str);
39    }
40}
```

Figura 3: Algoritmo de conversión de bases

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 sera de Binario a Decimal. Lo primero que se realizo fue buscar el punto decimal en la cadena binaria para futuras versiones. En este caso solo nos centraremos en los numeros 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

```
1 // Create needed objects and vars
2 GString *afterPoint = g_string_new(value -> str);
3 GString *beforePoint = g_string_new(value -> str);
4
5 gboolean flag = FALSE;
6 gfloat help = 0;
7 gint pot = 0;
8 gint result = 0;
9 // Separate the string from point
10 for (gint i = 0; i < value->len; i++) {
11     if (value -> str[i] == '.'){
12         beforePoint = g_string_truncate(beforePoint, i);
13         afterPoint = g_string_erase(afterPoint, 0, i + 1);
14         flag = TRUE;
15         break;
16     }
17 }
18 // Convert the part before at point
19 for (gint i = (beforePoint -> len) - 1; i >= 0; i--) {
20     if (beforePoint -> str[i] == '1') {
21         pot = (beforePoint->len - 1) - i;
22         help = pow(2, pot);
23         result += help;
24     }
25 }
```

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.

```
1 int bcdToDecimal(GString *value, int flag) {
2     GString *result = g_string_new("");
3     GString *trunc = g_string_new("");
4     GString *request = g_string_new("");
5     // Valid multipl to 4
6     gint difference = 0;
7     for (gint i = 4; i <= (value->len + 4); i += 4) {
8         if (i >= (value->len - 1)) {
9             difference = i - (value->len);
10        }
11    }
12    // Add char's for complete multi the 3
13    if (difference < 4 && difference != 0) {
14        for (gint i = 1; i <= difference; i++) {
15            g_string_prepend_c(value, '0');
16        }
17    }
18    for (gint i = 4; i < (value->len) + 4; i+=4) {
19        trunc = g_string_insert(trunc, 0, value->str);
20        // Recorta 4 numeros
21        trunc = g_string_erase(trunc, i, 0 - 1);
22        trunc = g_string_erase(trunc, 0, i - 4);
23        request = g_string_append_c(request, binToDecimal(trunc, 1));
24        // Vacía la cadena
25        trunc = g_string_erase(trunc, 0, 0 - 1);
26    }
27    g_print("Valor: %s\n", request->str);
28    if (flag == 0) {
29        gtk_entry_set_text(GTK_ENTRY(txtDec), request->str);
30    } else {
31        int retu = 0;
32        retu = atoi(request->str);
33        g_print ("return: %d\n", retu);
34        return retu;
35    }
36 }
```

Figura 5: Algoritmo de BCD a decimal

Para convertir de los demás formatos a BCD se utiliza un algoritmo que para de decimal a BCD, por lo que es posible reutilizarlo para las demás bases y así obtener su equivalente, el algoritmo se muestra en la figura 6

```
1 void decToBcd (int val) {
2     char str[20];
3     sprintf(str, "%d", val);
4     GString *value = g_string_new(str);
5     GString *request = g_string_new("");
6     int arg = 0;
7     GString *aux = g_string_new("");
8
9     for (int i = 0; i < (value->len); i++) {
10         // Cast gchar to int
11         arg = value->str[i] - '0';
12         if (arg == 0) {
13             aux = g_string_append(aux, "0000");
14         } else {
15             aux = g_string_append(aux, DecToHexaOctaDirect(arg, 2, 1));
16             g_print("pre aux: %s\n", aux->str);
17             if((aux->len - 1) < 4){
18                 for (int j = (aux->len); j < 4; j++) {
19                     aux = g_string_prepend_c(aux, '0');
20                 }
21             }
22         }
23         request = g_string_append(request, aux->str);
24         g_print("arg: %d request: %s aux: %s\n", arg, request->str, aux->str);
25         aux = g_string_erase(aux, 0, -1);
26     }
27     gtk_entry_set_text(GTK_ENTRY(txtBcd), request->str);
28 }
```

Figura 6: Algoritmo de decimal a BCD

4. Resultados

El programa actúa según lo esperado, siendo capaz de convertir de cualquier base a el resto (dentro de las admitidas. Figura 7). Aún es posible mejorar el código utilizando otras herramientas que nos provee C, además de agregar operaciones a el programa como lo son la suma, resta, etc. Pero dentro de lo esperado para este proyecto se cumplieron los Objetivos de la signación.

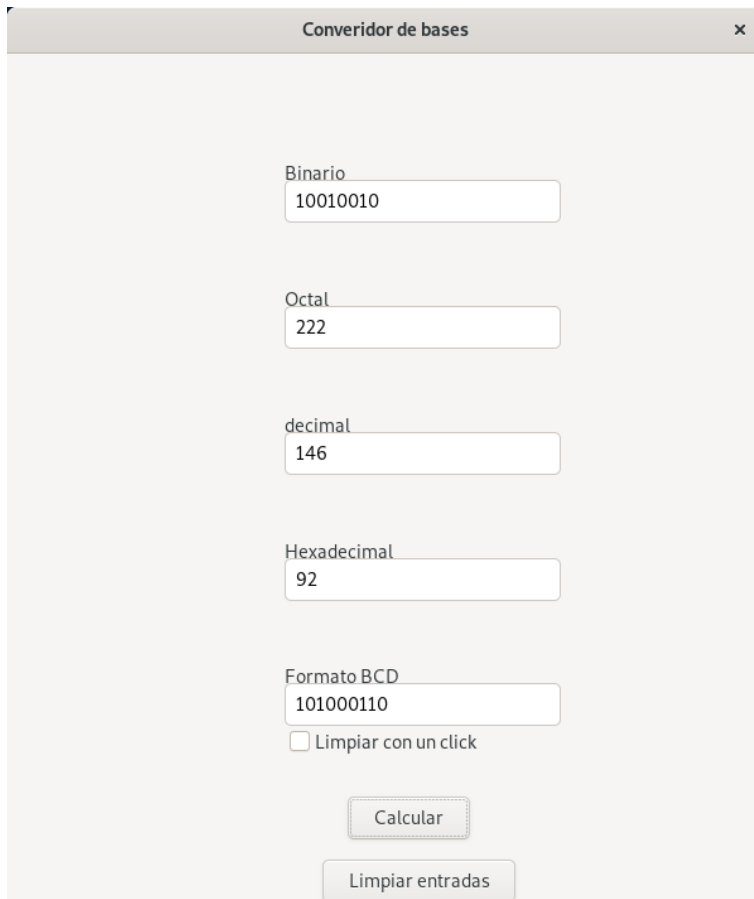


Figura 7: Muestra de funcion del programa finalizado

5. Conclusiones

El programa se completo con exito, 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 numero de la manera más eficiente posible, que es lo que se realizo en dicho algoritmo, reciclar y generalizar un algoritmo para mejorar su legibilidad y eficiencia.

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