ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

BÀI TẬP 2.04

Thành viên: Nguyễn Trung Nguyên – 20127404
Lê Đặng Minh Khôi – 20127213
Giảng viên: Thái Hùng Văn

MỤC LỤC

I. THÔNG TIN CHUNG

Nhóm gồm 2 thành viên:

Nguyễn Trung Nguyên – MSSV: 20127404

Lê Đặng Minh Khôi – MSSV: 20127213

II. NỘI DUNG THỰC HIỆN

1. Giới thiệu

Chương trình được thực hiện theo yêu cầu của thầy Thái Hùng Văn, giảng viên bộ môn Hệ thống máy tính - Khoa Công nghệ thông tin – Trường Đại học Khoa học Tự nhiên.

Công cụ sử dụng: Visual Studio 2019

Hỗ trợ tạo giao diện: MFC

2. Mục tiêu chương trình

Chương trình cho phép chuyển đổi từ số thập phân sang chuỗi bit ở dạng số thực 32/64bit và số nguyên 8/16/32bit (không dấu và có dấu) và ngược lại.

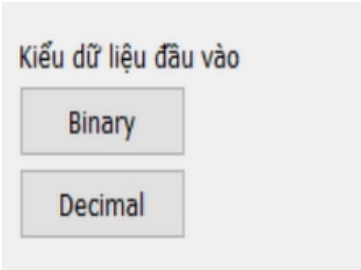
3. Phân chia công việc và mức độ hoàn thành

Công việc	Người thực hiện	Mức độ hoàn thành	Ghi chú
Chuyển đổi số thực 32/64bit	Lê Đặng Minh Khôi	100%	
Chuyển đổi số nguyên sang 8/16/32bit	Lê Đặng Minh Khôi	100%	
Chuyển đổi dãy 8/16/32bit sang định dạng số nguyên	Nguyễn Trung Nguyên	100%	
Chuyển đổi dãy 32/64bit sang dạng số thực	Lê Đặng Minh Khôi	100%	
Thiết kế giao diện GUI	Nguyễn Trung Nguyên	100%	
Viết báo cáo	Nguyễn Trung Nguyên	100%	

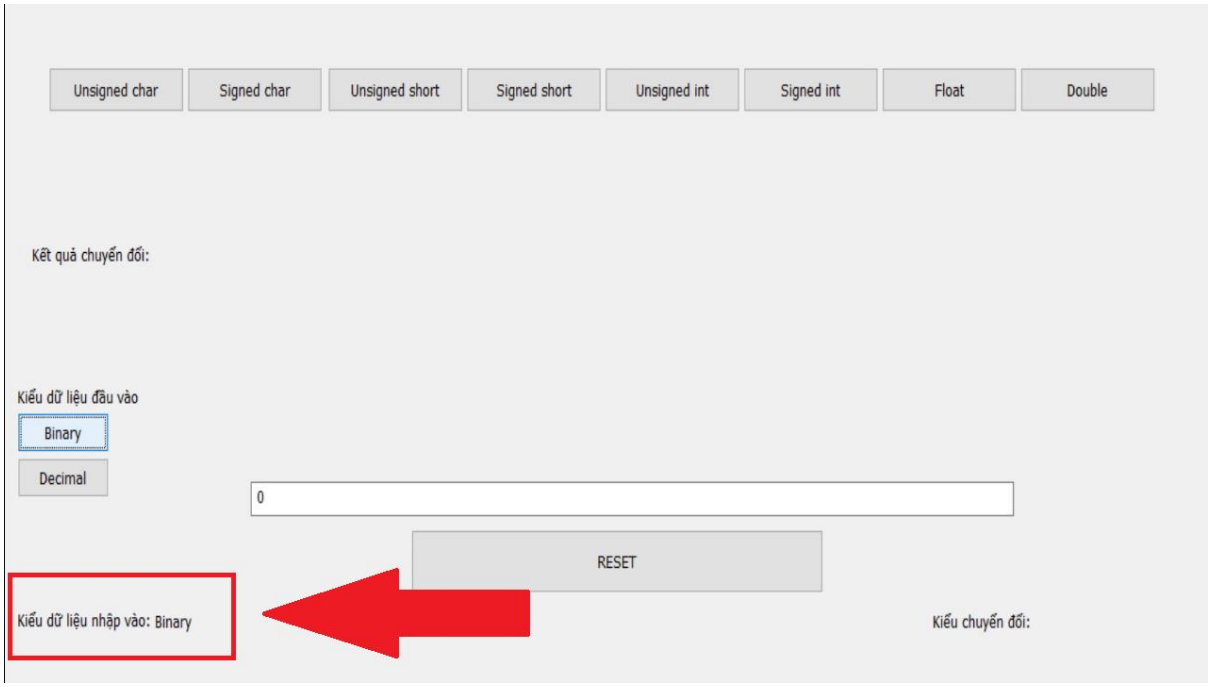
4. Một số lưu ý

4.1. Hướng dẫn sử dụng chương trình.

Khi chương trình được mở ta tiến hành chọn kiểu dữ liệu mà ta muốn nhập vào bằng cách bấm vào nút trong ô ***Kiểu dữ liệu đầu vào***:



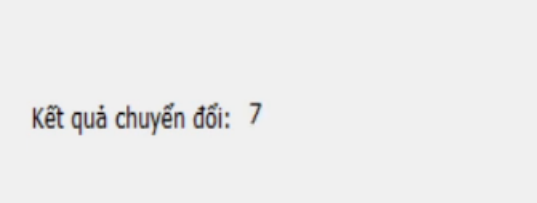
Sau khi chọn chương trình sẽ hiển thị kết quả ở góc phía trái dưới cùng của cửa sổ:

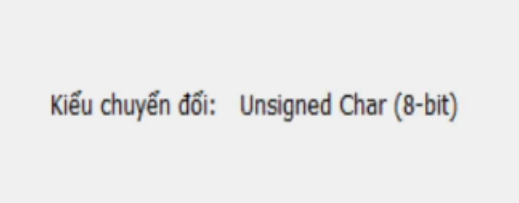


Sau đó ta tiến hành nhập giá trị mà ta muốn chuyển đổi:

Sau khi nhập ta sẽ chọn kiểu mà mình muốn chuyển đổi bằng cách nhấn vào các nút có kiểu chuyển đổi tương ứng. Ví dụ ta muốn chuyển dãy bit 111 ở dạng unsigned char sang dạng thập phân thì ta click vào nút ***Unsigned char***.

Sau khi chọn kết quả sẽ được hiển thị ở mục ***Kết quả chuyển đổi***. Ngoài ra người dùng cũng có thể kiểm tra lại sự lựa chọn của mình ở mục ***Kiểu dữ liệu chuyển đổi***.





Việc này được áp dụng tương tự với các kiểu chuyển đổi khác. Khi muốn xóa kiểu chuyển đổi, kết quả hiển thị cũng như số đã nhập vào, người dùng chỉ việc click vào nút **Reset** phía dưới ô nhập.

4.2. Mã nguồn

Các hàm xử lí đối với số nhị phân:

https://onlinegdb.com/PMj8QUrs6

Các hàm xử lí đối với số nguyên:

https://onlinegdb.com/fv-Mqf1x3

Các hàm xử lý với số thực theo dạng số thực chuẩn IEEE754:

https://onlinegdb.com/o-Ws39-_U

5. Mã nguồn và mô tả thuật toán

5.1. Chuyển đổi chuỗi 32bit sang dạng số thực chính xác đơn

a. Mã nguồn:

```
float ieee754::binary_to_float(const std::string& value) {
    this->_bin_value = value;
    unsigned int sign = (_bin_value[0] == '1'), exponnet =
_operator.decimal_get(value, 1, 8);
    std::string  mantissa = _sub_operator.strip(_bin_value, 9, 31);
    float out = 1;
    int str_len = mantissa.length();
    for (int i = 0; i < str_len; i++) {
        out += (mantissa[i] == '1' ? 1 : 0) * pow(2, -(i + 1));
    }
    out *= pow(2, exponnet - 127);
    return (sign ? -out : out);
}
```

b. Mô tả thuật toán:

Thuật toán trên bao gồm các bước như sau:

1. Trích xuất bit dấu: nếu bit đầu tiên là bit 1 thì kết quả là âm, ngược lại là dương.

2. Trích xuất số mũ (Exponent): 8 bit tiếp theo trong chuỗi là mã hoá của số mũ, ta tiến hành lấy 8 bit này và chuyển thành dạng số thập phân.

3. Kết quả có được sau khi thực hiện bước 2 bị lệch một khoảng 127. Chính vì thế sau khi chuyển đổi ta sẽ phải trừ giá trị đó cho 127.

4. Chuyển đổi phần định trị (Mantissa) sang dạng thập phân: 23 bits cuối của chuỗi mã hoá cho phần định trị. Ta sẽ nhân bit ngoài cùng bên trái cho 2^{-1} và cộng cho giá trị của bit tiếp theo nhân với 2^{-2} cứ như thế cho bit cuối cùng sẽ là 2^{-n}.

5. Ta xác định kết quả dạng số thực theo công thức: *(1 + mantissa) * 2^(unbiased_exponent)*

6. Cuối cùng ta xác định dấu của số thực vừa tính được bằng cách xét bit dấu đã xác định ở bước 1 (1 – số âm, 0 – số dương) [1]

5.2. Chuyển đổi từ số thực chính xác đơn sang chuỗi 32bit

a. Mã nguồn:

```
std::string ieee754::float_to_binary(const float& value) {
    this->_i32_value = value;
    std::string temp = _operator.binary_get(_i32_value);
    //get exponent
    int iter = 0; int length = temp.length();
    while (temp[iter] != '.' && iter < length) {
        iter++;
    }
    int exponent = iter - 1 + 127;
    // generate ieee 754
    std::stringstream out; out << (_i32_value < 0) <<
_operator.binary_get(exponent, 8);
    //add mantissa
    iter = 1;
    while (iter < length) {
        if (temp[iter] == '.') { iter++; continue; }
        out << temp[iter];
        iter++;
    }
    std::string result = out.str();

    if (result.length() < 32)
        while (result.length() != 32) { out << 0; result = out.str(); }
    else
        if (result.length() > 32) result.erase(result.begin() + 32,
result.end());

    return result;
}
```

Mô tả thuật toán:

Thuật toán gồm những bước như sau:

1. Xác định bit đầu thông qua dấu (âm - 1, dương – 0)

2. Chuyển đổi phần nguyên thành chuỗi nhị phân không dấu.

3. Chuyển đổi phần thập phân thành chuỗi nhị phân

4. Chuẩn hóa giá trị thông qua việc điều chỉnh số mũ.

5. Cộng thêm phần chênh lệch (bias) vào số mũ.

6. Chuyển đổi phần số mũ đó thành chuỗi nhị phân không dấu.

7. Xác định các bit cuối cùng cho phần định trị (Mantissa).

8. Gộp các chuỗi bit lại với nhau. [1]

5.3. Chuyển đổi chuỗi 64bit sang dạng số thực có dạng chính xác kép

a. Mã nguồn:

```
double ieee754::binary_to_double(const std::string& value) {
    this->_bin_value = value;
    unsigned int sign = (_bin_value[0] == '1'), exponnet =
_operator.decimal_get(value, 1, 11);
    std::string  mantissa = _sub_operator.strip(_bin_value, 12, 63);
    double out = 1;
    int str_len = mantissa.length();
    for (int i = 0; i < str_len; i++) {
        out += (mantissa[i] == '1' ? 1 : 0) * pow(2, -(i + 1));
    }
    double pow_value = pow(2, exponnet - 1023);
    out *= pow_value;
    return (sign ? -out : out);
}
```

b. Mô tả thuật toán:

Thuật toán bao gồm các bước thực hiện như sau:

1. Trích xuất bit dấu: nếu bit đầu tiên là bit 1 thì kết quả là âm, ngược lại là dương.

2. Trích xuất số mũ (Exponent): 11 bit tiếp theo trong chuỗi là mã hoá của số mũ, ta tiến hành lấy 11 bit này và chuyển thành dạng số thập phân.

3. Kết quả có được sau khi thực hiện bước 2 bị lệch một khoảng 1023. Chính vì thế sau khi chuyển đổi ta sẽ phải trừ giá trị đó cho 1023.

4. Chuyển đổi phần định trị (Mantissa) sang dạng thập phân: 52 bits cuối của chuỗi mã hoá cho phần định trị. Ta sẽ nhân bit ngoài cùng bên trái cho 2^{-1} và cộng cho giá trị của bit tiếp theo nhân với 2^{-2} cứ như thế cho bit cuối cùng sẽ là 2^{-n}.

5. Ta xác định kết quả dạng số thực theo công thức: *(1 + mantissa) * 2^(unbiased_exponent)*

6. Cuối cùng ta xác định dấu của số thực vừa tính được bằng cách xét bit dấu đã xác định ở bước 1 (1 – số âm, 0 – số dương)

5.4. Chuyển đổi từ số thực chính xác kép sang chuỗi 64bit

a. Mã nguồn:

```
std::string ieee754::double_to_binary(const double& value) {
    this->_i64_value = value;
    std::string temp = _operator.binary_get(_i64_value);
    //get exponent
    int iter = 0; int length = temp.length();
    while (temp[iter] != '.' && iter < length)
        iter++;

    int exponent = iter - 1 + 1023;
    // generate ieee 754
    std::stringstream out; out << (_i64_value < 0) <<
_operator.binary_get(exponent, 11);
    //add mantissa
    iter = 1;
    while (iter < length) {
        if (temp[iter] == '.') { iter++; continue; }
        out << temp[iter];
        iter++;
    }
    std::string result = out.str();
    if (result.length() < 64)
        while (result.length() != 64) { out << 0; result = out.str(); }
    else
        if (result.length() > 64) result.erase(result.begin() + 64,
result.end());

    return result;
}
```

b. Mô tả thuật toán:

Thuật toán gồm những bước như sau:

1. Xác định bit đầu thông qua dấu (âm - 1, dương – 0)

2. Chuyển đổi phần nguyên thành chuỗi nhị phân không dấu.

3. Chuyển đổi phần thập phân thành chuỗi nhị phân

4. Chuẩn hóa giá trị thông qua việc điều chỉnh số mũ.

5. Cộng thêm phần chênh lệch (bias) vào số mũ.

6. Chuyển đổi phần số mũ đó thành chuỗi nhị phân không dấu.

7. Xác định các bit cuối cùng cho phần định trị (Mantissa).

8. Gộp các chuỗi bit lại với nhau.

5.5. Chuyển đổi chuỗi 8/16/32bit sang dạng số nguyên không dấu

a. Mã nguồn:

```
unsigned int integer::binary_to_unsigned_int(const std::string& value, int
bits) {
    this->_bin_value = value;
    int result = _operator.decimal_get(_bin_value, 0, bits - 1);
    return result;
}
```

b. Mô tả thuật toán:

Để chuyển đổi từ chuỗi bit sang dạng số thập phân không dấu ta lấy tổng giá trị từng bit (từ trái qua) nhân với 2^{n-1}, giá trị của $n-1$ sẽ giảm 1 đơn vị khi đem nhân với bit tiếp theo.

VD: $10_2 = 1*2^1 + 0*2^0 = 2_{10}$

5.6. Chuyển đổi 8/16/32bit sang dạng số nguyên có dấu

Mã nguồn:

* 8-bit:

```
int8_t integer::binary_to_char(const std::string& value) {
    this->_bin_value = value;
    if (_bin_value[0] == '1') _bin_value =
_operator.twos_complement(_bin_value);
    int result = _operator.decimal_get(_bin_value, 1, 7);
    return (_bin_value[0] == '1' ? -result : result);
}
```

* 16-bit:

```
int16_t integer::binary_to_short(const std::string& value) {
    this->_bin_value = value;
    if (_bin_value[0] == '1') _bin_value =
_operator.twos_complement(_bin_value);
    int result = _operator.decimal_get(_bin_value, 1, 15);
    return (_bin_value[0] == '1' ? -result : result);
}
```

* 32-bit:

```
int integer::binary_to_integer(const std::string& value) {
    this->_bin_value = value;
    if (_bin_value[0] == '1') _bin_value =
_operator.twos_complement(_bin_value);
    int result = _operator.decimal_get(_bin_value, 1,31);
    return (_bin_value[0] == '1' ? -result : result);
}
```

Mô tả thuật toán:

Các thuật toán chuyển đổi từ chuỗi bit sang số thập phân có dấu cũng giống như việc chuyển đổi các chuỗi bit sang số thập phân không dấu, chỉ khác là thêm 1 bước kiểm tra dấu của chuỗi. Nếu bit leftmost là 1 thì tiến hành bù 2 chuỗi bit trước khi chuyển đổi, ngược lại thì ta tiến hành chuyển đổi bình thường.

5.7. Chuyển đổi từ dạng số nguyên không dấu sang chuỗi bit 8/16/32bit:

a. Mã nguồn:

```
std::string integer::unsigned_int_to_binary(const int& value, int bits) {
    this->_i32_value = value;
    if (_i32_value < 0) _i32_value = 0;
    std::string bin = _operator.binary_get(_i32_value, bits);
    return bin;
}
```

b. Mô tả thuật toán: Để xác định số chuỗi bit từ 1 số nguyên không dấu ta tiến hành:

- Chia lặp đi lặp lại số đó cho 2 (Phép chia dừng lại khi kết quả lần chia cuối cùng bằng 0).

- Lấy các số dư theo chiều đảo ngược sẽ được số nhị phân cần tìm.

5.8. Chuyển đổi từ dạng số nguyên sang chuỗi bit 8/16/32bit:

a. Mã nguồn:

* 8-bit:

```
std::string integer::char_to_binary(const int8_t& value) {
    this->_i8_value = value;
    std::string bin = _operator.binary_get(_i8_value, 8);
    return bin;
}
```

* 16-bit:

```
std::string integer::short_to_binary(const int16_t& value) {
    this->_i16_value = value;
    std::string bin = _operator.binary_get(_i16_value, 16);
    return bin;
}
```

* 32-bit:

```
std::string integer::integer_to_binary(const int& value) {
    this->_i32_value = value;
    std::string bin = _operator.binary_get(_i32_value, 32);
    return bin;
}
```

b. Mô tả thuật toán: Việc chuyển đổi cũng tương tự như đối với phần số thập phân không dấu.

III. TÀI LIỆU THAM KHẢO

[1] K. Dewey, "Converting Between Decimal and Binary Floating-Point Numbers," [Trực tuyến]. Available: https://kyledewey.github.io/comp122-fall17/lecture/week_2/floating_point_interconversions.html?fbclid=IwAR0dGX6Um2F_bvuVaJ4cfdH97lS3BOTgkw80oelP9rOdw6xrV7fTg-GgUVU. [Đã truy cập 12 11 2021].

[2] "Wikipedia," [Trực tuyến]. Available: https://en.wikipedia.org/wiki/Double-precision_floating-point_format. [Đã truy cập 12 11 2021].

[3] Windstar981, "Codelearn," [Trực tuyến]. Available: https://codelearn.io/sharing/cach-chuyen-doi-cac-he-so-dem. [Đã truy cập 12 11 2021].

[4] C. T. Bug, "YouTube," [Trực tuyến]. Available: https://www.youtube.com/watch?v=8bbC2R6vICY&ab_channel=Codeto%C3%A0nbug. [Đã truy cập 12 11 2021].