**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CNTT**



**KIẾN TRÚC MÁY TÍNH VÀ HỢP NGỮ**

**ĐỒ ÁN 1**

**BIỂU DIỄN VÀ TÍNH TOÁN SỐ NGUYÊN LỚN**

***Thành viên* : Nguyễn Hoàng Lưu 1612364**

**Phạm Hoàng Minh 1612380**

**Huỳnh Nguyễn Nhật Minh 1612382**

**Nguyễn Hoàng Hiếu Nghĩa 1612420**

**Nguyễn Thái Hoàng 1412180**

***Lớp* : KTMT & HN CQ2016/1 (Ca chiều thứ 6)**

***Giảng viên* : Lê Viết Long**

## **MỤC LỤC**

1. [1. Thông tin thành viên và công việc : 3](#_Toc510209649)
2. [2. Ý tưởng và thiết kế : 4](#_Toc510209650)

[I. Môi trường lập trình : 4](#_Toc510209651)

[II. Ý tưởng và thiết kế : 4](#_Toc510209652)

1. [3. Đánh giá: 7](#_Toc510209653)
2. [4. Tài liệu tham khảo: 8](#_Toc510209654)

# Thông tin thành viên và công việc :

|  |  |
| --- | --- |
| **Thành viên** | **Công việc** |
| **Nguyễn Hoàng Lưu**  **1612364** | * Tìm tài liệu * Lập trình các hàm hỗ trợ * Viết báo cáo |
| **Phạm Hoàng Minh**  **1612380** | * Lập trình hàm QIntToDec, DecToQInt, toán tử AND, OR, XOR, NOT * Phân công công việc * Bổ sung báo cáo |
| **Huỳnh Nguyễn Nhật Minh**  **1612382** | * Toán tử + - \* /, lập trình hàm BinToHex, HexToBin * Tổng hợp code * Bổ sung báo cáo |
| **Nguyễn Hoàng Hiếu Nghĩa**  **1612420** | * Toán tử << >> * Tester * Viết báo cáo |
| **Nguyễn Thái Hoàng**  **1412180** | * Lập trình hàm nhập xuất (ScanQInt, PrintQInt) * Bổ sung báo cáo |

# Ý tưởng và thiết kế :

## Môi trường lập trình :

* Microsoft Visual Studio 2013
* Ngôn ngữ C++

## Ý tưởng và thiết kế :

* Sử dụng 4 mảng nhỏ, mỗi mảng 4 bytes để biểu diễn số Qint.
* Xây dựng các hàm tính toán của QInt: chuyển đổi qua lại giữa các hệ cơ số (Binary, Decimal, Hexadecimal), các toán tử + - \* / =, các phép xử lí bit AND (&) OR (|) XOR (^) NOT (~), các toán tử dịch trái (<<) và dịch phải (>>).
* Trong class QInt ta có các hàm phụ như:
* getBit(i): lấy bit tại vị trí thứ i
* turnOn(i): bật bit tại vị trí thứ i lên 1
* turnOff(i): tắt bit tại vị trí thứ i xuống 0
* getOneComplement(): lấy số bù 1
* getTwoComplement(): lấy số bù 2
* DivDec(Dec): chia chuỗi cho 2
* CharToInt(c): chuyển đổi kí tự c thành số nguyên
* **Các hàm chuyển hệ cơ số:** ta chuyển từ chuỗi số ban đầu ra QInt bằng cách lưu các bit vào 4 mảng nhỏ. Sau đó từ số QInt đã có ta chuyển về chuỗi theo hệ cơ số mong muốn và xuất ra màn hình để hiển thị.
  + **Chuyển từ chuỗi Dec 🡪 QInt**: tạo biến QInt = 0 và biến sign = 1 để lưu dấu của chuỗi. Bắt đầu vòng lặp từ i = 0 (i<=127). Lấy kí tự cuối cùng của chuỗi chuyển sang dạng số rồi chia lấy dư cho 2. Nếu số dư = 1 thì bật bit thứ i của QInt lên, sau đó chia chuỗi số cho 2 lấy phần nguyên và tăng i lên 1. Sau khi kết thúc vòng lặp, nếu chuỗi vẫn còn giá trị và khác 0 thì chuỗi nhập vào lớn hơn 128 bits và in ra màn hình “Math Error”. Nếu chuỗi ban đầu có dấu “−” thì ta xóa dấu trừ và chuyển về QInt như trên, sau khi kết thúc thì ta lấy bù 2 của QInt để biến QInt thành số âm.
  + **Chuyển đổi từ QInt 🡪 Dec**: ta chạy biến i từ 0 đến 127. Ở mỗi vòng lặp ta lấy số QInt chia cho 10, rồi lấy một biến int chứa 4 bits đầu của số dư rồi gán biến đó lên đầu chuỗi kết quả, sau đó gán số QInt bằng phần nguyên của phép chia rồi tiếp tục. Nếu QInt là số âm (bit thứ 127 = 1) thì ta lấy bù 2 của QInt để chuyển sang chuỗi, sau đó gán kí tự “−” vào đầu chuỗi.
  + **Chuyển đổi từ Bin 🡪 QInt**: Duyệt từng kí tự của chuỗi Bin từ trái sang cho đến hết, với mỗi kí tự, nếu kí tự đó là ‘0’ thì ta tắt bit tương ứng của Qint, ngược lại nếu bằng ‘1’ ta sẽ bật bit tương ứng của Qint.
  + **Chuyển đổi từ QInt 🡪 Bin**: Duyệt từng bit của QInt từ trái sang cho đến hết, với bit, nếu bit đó là 0 thì ta gán kí tự tương ứng tại vị trí của chuỗi Bin bằng ‘1’, ngược lại nếu bit bằng 0 ta sẽ gán kí tự tương ứng của chuỗi Bin bằng ‘0’.
  + **Chuyển đổi từ Bin 🡪 Hex**: Mỗi lần ta xét 4 kí tự của chuỗi Bin (tương ứng 4 bit), chuyển 4 kí tự đó thành mã tương ứng trong hệ 16 và gán vào chuỗi Hex.
  + **Chuyển đổi từ Hex 🡪 Bin**: Xét từng kí tự trong chuỗi Hex, với mỗi kí tự đó ta chuyển thành mã tương ứng trong hệ 2 (4 bit). Lặp đến hết ta sẽ được kết quả là một chuỗi nhị phân tương ứng với chuỗi Hex ban đầu.
* **Phép gán “ = ”**: set lại các bit theo biến cần gán
* **Phép cộng “ + ”**: Chuyển về hệ cơ số 2 rồi cộng từng bit của 2 số, sử dụng 1 biến tạm để nhớ sang vị trí tiếp theo.
* **Phép trừ “ - ”**: ta xem a – b = a + (-b). Chuyển về hệ cơ số 2 rồi lấy số bù 2 của b, sau đó dùng toán tử cộng a với số bù 2 của b.
* **Phép nhân “ \* ”**: Sử dụng thuật toán Booth: ta xét M \* Q, khởi tạo A=0, k=n (số bit của Q), q1=0 (biến sẽ chứa bit cuối cùng của Q trước khi dịch phải số học). Lặp khi k>0 nếu bit cuối của Q và q1 là 1 0 thì gán A=A-M hoặc 0 1 thì gán A=A+M sau đó dùng q1 lưu lại biến cuối của Q rồi dịch phải số học A và Q, k giảm 1 đơn vị. Cuối cùng khi thoát khỏi vòng lặp ta trả về Q. Lý do không trả về giá trị nối giữa A&Q là do số bit ban đầu ta quy định là 128, sau khi dịch 128 lần, tất cả các bit giá trị sẽ được lưu trong Q.
* **Phép chia “ / ”**: ta xét Q/M (M<Q), khởi tạo A=128 bit 0 nếu Q>0, A=128 bit 1 nếu Q<0, k=128. Lặp khi k>0. Trước tiên, ta dịch trái luận lý dãy [A,Q] bằng cách ta dịch trái luận lý A trước rồi sau đó gán bit thứ 127 của Q vào bit thứ 0 của A, rồi sau đó ta dịch trái luận lý Q. Sau đó ta gán A = A-M và xét 2 trường hợp:   
  + TH1: A < 0, ta gán bit 0 của Q bằng 0 và A = A+M.   
  + TH2: ngược lại, ta gán bit 0 của Q bằng 1.   
  Tiếp theo ta giảm k đi 1 đơn vị và lặp đến khi nào k = 0 thì dừng vòng lặp. Kết quả của phép chia lấy nguyên là Q.
* **Phép chia lấy dư “ % ”**: thuật toán tương tự như phép chia lấy nguyên, kết quả của phép chia lấy dư là A.
* **Toán tử AND ( & )**: thực hiện trên nhị phân, AND từng bit của 2 số đó rồi gán vào kết quả.
* **Toán tử OR ( | )**: thực hiện trên nhị phân, OR từng bit của 2 số rồi gán vào kết quả.
* **Toán tử XOR ( ^ )**: thực hiện trên nhị phân, XOR từng bit của 2 số rồi gán vào kết quả.
* **Toán tử NOT ( ~ )**: thực hiện trên nhị phân, NOT từng bit của số đó rồi gán vào kết quả.
* **Phép dịch trái ( << )**: thực hiện trên nhị phân, dịch trái theo số lượng i bit muốn dịch rồi gán vào kết quả.
* **Phép dịch phải ( >> )**: thực hiện trên nhị phân, dịch phải theo số lượng i bit muốn dịch rồi gán vào kết quả.
* Phạm vi biểu diễn: -2127 🡪 2127 – 1

1. **Đánh giá:**

* **Các chức năng thực hiện được**: các hàm chuyển đổi giữa các hệ cơ số Binary, Decimal, Hexadecimal, các toán tử + - \* / =, các phép tính AND, OR, XOR, NOT, và toán tử dịch trái <<, dịch phải >>.
* **Các chức năng chưa thực hiện được**: thiết kế giao diện cho đồ án.
* **Tổng đánh giá mức độ đồ án**: Nhóm tự đánh giá hoàn thành đầy đủ yêu cầu của đồ án ngoại trừ yêu cầu thiết kế giao diện cho đồ án.

1. **Tài liệu tham khảo**:

* Slide bài giảng trên moodle của lớp KTMT & HN CQ2016/1 (Ca chiều thứ 6).
* Các trang web youtube, google,…
* Các diễn đàn thảo luận về code trên web.
* Các nhóm thảo luận về code trên mạng xã hội.