MINISTRY OF EDUCATION AND TRAINING

HO CHI MINH UNIVERSITY OF SCIENCE

FACULTY OF INFORMANTION TECHNOLOGY _ K20

Data Structures and Algorithms

PROJECT

ARITHMETIC EXPRESSION CALCULATION

Theory lecturer: Ngô Minh Nhựt

Practical lecturer: Bùi Huy Thông, Trần Thị Thảo Nhi

Student:

Trần Tiến Hoàng – 20127424

Lâm Kim Nhân – 20127579

Lê Võ Huỳnh Thanh - 20127072

Class: 20CLC09

Mục Lục

PHÂN CÔNG CÔNG VIỆC

Trần Tiến Hoàng – 20127424: Lập trình các hàm: string postfix(string s), int Rank(char c), float calculation(string s).

Lâm Kim Nhân – 20127579: Lập trình hàm main, trình bày nghiên cứu 1.1 cách chuyển đổi biểu thức trung tố sang hậu tố.

Lê Võ Huỳnh Thanh – 20127072: Lập trình hàm bool CheckExpression(string s), trình bày nghiên cứu 1.1 cách chuyển đổi biểu thức trung tố sang tiền tố và biểu thức tiền tố sang hậu tố, trình bày báo cáo đồ án.

CHAPTER 1: TỔNG QUAN

I. Hoàn thành

1.1 Nghiên Cứu

- Chuyển đổi biểu thức trung tố thành tiền tố
- Chuyển dổi biểu thức trung tố thành hậu tố
- Chuyển dổi biểu thức tiền tố thành hậu tố

1.2 Lập Trình

- Hoàn thành chương trình chuyển đổi biểu thức trung tố về hậu tố
- Xuất ra kết quả là số thực từ biểu thức hậu tố

II. Những việc đã học được

- Kĩ năng làm việc nhóm.
- Hiểu được cách chuyển biểu thức trung tố sang tiền tố, hậu tố hay từ tiền tố sang hậu tố.
- Biết thêm các thư viện mới (stack, iomanip,..).

CHAPTER 2: CẤU TRÚC THUẬT TOÁN

1.1 Nghiên cứu

- **Chuyển đổi biểu thức trung tố thành biểu thức tiền tố:**

Thuật toán:

Khởi tạo một stack rỗng và vị trí i ban đầu bằng 0.

Khi biểu thức đầu vào là biểu thức đúng, đảo ngược biểu thức trung tố lại sau đó bắt đầu duyệt i, lấy giá trị kí tự tại vị trí i:

- Nếu là toán hạng: output.
- Nếu là dấu đóng ngoặc “)“ thì cho vào stack
- Nếu là dấu mở ngoặc “(” ta sẽ lấy các toán tử trong stack ra và cho vào output cho đến khi gặp dấu đóng ngoặc “)“. Lúc này dấu ngoặc sẽ được đưa ra khỏi stack nhưng không xuất hiện ở ouput.
- Nếu là toán tử: Chừng nào ở đỉnh stack là toán tử và toán tử đó có độ ưu tiên lớn hơn hoặc bằng toán tử hiện tại thì lấy toán tử đó ra khỏi stack và cho vào output. Sau đó đưa toán tử hiện tại vào stack. Khi đã duyệt hết biểu thức infix, nếu trong stack còn phần tử thì lấy các giá trị kí tự trong đó ra và cho lần lượt vào output.
- Cuối cùng là đảo ngược biểu thức một lần nữa và ta sẽ thu được kết quả.

Ví dụ

- Chuyển biểu thức trung tố ((A+B)*C)/(D-E) biểu thức hậu tố:
  Đầu tiên đảo ngược biểu thức trung tố)E-D(/)C*)B+A((sau đỏ chuyển sang hậu tố

Infix	**Stack**	**Prefix**
)E-D(/)C*)B+A((	{Empty}	**{Empty}**
E-D(/)C*)B+A((	)	**{Empty}**
-D(/)C*)B+A((	)	**E**
D(/)C*)B+A((	)-	**E**
(/)C*)B+A((		**ED-**

/)C*)B+A((		**ED-**
)C*)B+A((	/	**ED-**
C*)B+A((	/)	**ED-**
***)B+A((**	/)	**ED-C**
)B+A((	/)*	**ED-C**
B+A((	/)*)	**ED-C**
+A((	/)*)	**ED-CB**
A((	/)*)+	**ED-CB**
((	/)*)+	**ED-CBA**
(	/)*	**ED-CBA+**
{Empty}	/	**ED-CBA+***
{Empty}	{Empty}	**ED-CBA+*/**
{Empty}	{Empty}	**/*+ABC-DE**

Sau đó đảo ngược biểu thức đã chuyển sẽ ra được biểu thức đổi từ trung tố sang tiền tố.

- **Chuyển đổi biểu thức trung tố thành biểu thức hậu tố:**

Thuật toán:

Khởi tạo một stack rỗng và vị trí i ban đầu bằng 0.

Khi biểu thức đầu vào là biểu thức đúng, bắt đầu duyệt i, lấy giá trị kí tự tại vị trí i:

- Nếu là toán hạng: output.
- Nếu là dấu mở ngoặc "(" thì cho vào stack
- Nếu là dấu đóng ngoặc ")" ta sẽ lấy các toán tử trong stack ra và cho vào output cho đến khi gặp dấu mở ngoặc "(". Lúc này dâú ngoặc sẽ được đưa ra khỏi stack nhưng không xuất hiện ở ouput.

- Nếu là toán tử: Chừng nào ở đỉnh stack là toán tử và toán tử đó có độ ưu tiên lớn hơn hoặc bằng toán tử hiện tại thì lấy toán tử đó ra khỏi stack và cho vào output. Sau đó đưa toán tử hiện tại vào stack. Khi đã duyệt hết biểu thức infix, nếu trong stack còn phần tử thì lấy các giá trị kí tự trong đó ra và cho lần lượt vào output.

Ví dụ:

Chuyển biểu thức trung tố A+(B-C)*D+E/(F*G) sang biểu thức hậu tố:

Infix	**Stack**	**Postfix**
A+(B-C)*D+E/(F*G)	{Empty}	**{Empty}**
+(B-C)*D+E/(F*G)	A	**{Empty}**
(B-C)*D+E/(F*G)	+	**A**
B-C)*D+E/(F*G)	+(	**A**
-C)*D+E/(F*G)	+(	**AB**
C)*D+E/(F*G)	+(-	**AB**
)*D+E/(F*G)	+(-	**ABC**
***D+E/(F*G)**	+	**ABC-**
D+E/(F*G)	+*	**ABC-**
+E/(F*G)	+*	**ABC-D**
E/(F*G)	+	**ABC-D*+**
/(F*G)	+	**ABC-D*+E**
(F*G)	+/	**ABC-D*+E**
F*G)	+/(	**ABC-D*+E**
***G)**	+/(	**ABC-D*+EF**
G)	+/(*	**ABC-D*+EF**

)	+/(*	**ABC-D*+EFG**
{Empty}	+/	**ABC-D*+EFG***
{Empty}	{Empty}	**ABC-D*+EFG*/+**

- **Chuyển đổi biểu thức tiền tố thành biểu thức hậu tố:**

Đọc kí tự từ trái qua phải. Nếu là toán tử ta liên kết toán tử đó với số nguyên 2 (dùng pair trong c++) và đẩy pair đó vào stack. Nếu là toán hạng thì ta output toán hạng và lấy pair trên cùng của stack và giảm số nguyên đi 1. Nếu pair trên cùng stack có số nguyên = 0 thì lấy pair đó ra khỏi stack và output toán tử. Khi chạy hết toàn bộ kí tự trong biểu thức tiền tố, ta lần lượt output ra từng kí tự còn lại trong stack.

Ví dụ: chuyển biểu thức tiền tố sau sang hậu tố + x – y z

Prefix	**Stack**	**Postfix**
+ x – y z	{Empty}	**{Empty}**
x – y z	{+,2}	**{Empty}**
– y z	{+,1}	**x**
Y z	{+,1} , {-,2}	**x**
Z	{+,1},{-,1}	**x y**
{Empty}	{+,1}	**x y z -**
{Empty}	{Empty}	**x y z - +**

1.2 Lập Trình

- Hàm int Rank(c) trả về độ ưu tiên của toán tử, toán hạng c.
- Thuật toán:

Kiểm tra kí tự c nếu là ‘ ‘ sẽ cho độ ưu tiên 0, số nguyên từ 0 đến 9 hoặc kí tự “.” sẽ cho độ ưu tiên 1, phép cộng – trừ sẽ cho độ ưu tiên 2, phép nhân – chia sẽ cho độ ưu tiên 3, dấu “(“ sẽ cho độ ưu tiên 4 và dấu “)” sẽ cho độ ưu tiên 5.

```
int Rank(char c)
{
if (c == ' ')
            return 0;
if ((c >= '0' && c <= '9') || c == '.')
            return 1;
if (c == '+' || c == '-')
            return 2;
if (c == '*' || c == '/' || c == '^')
```

```
            return 3;
if (c == '(')
            return 4;
if (c == ')')
            return 5;
}
```

Hàm bool CheckExpression(string s) trả về true nếu string s là phương trình hợp lí, false nếu không và sử dụng hàm int Rank(c) để phục vụ.

- Ta có các quy tắc sau để kiểm tra tính hợp lệ của một phép tính:

+ Số lượng "mở ngoặc" luôn lớn hơn hoặc bằng số lượng "đóng ngoặc" tính từ kí tự đầu đến kí tự bất kì và bằng nhau nếu tính từ đầu đến cuối chuỗi.
+ Không có trường hợp 2 toán tử hoặc 2 toán hạng nằm cạnh nhau.
+ Sau "mở ngoặc" phải là toán hạng.
+ Sau "đóng ngoặc" phải là toán tử hoặc kết thúc chuỗi.

- Thuật toán:

+ Vòng lặp chạy từ 0 đến độ dài của chuỗi s.
+ Braket1,2,3 = số lượng "mở ngoặc" - "đóng ngoặc" => nếu braket < 0 thì trả về false.
+ Biến check lưu rank của toán tử/toán hạng trước đó.
+ So sánh rank toán tử/toán hạng trước đó với toán tử/toán hạng hiện tại, nếu vi phạm các quy tắc ở trên thì trả về false.
+ Kết thúc mỗi vòng lặp, cập nhật biến check bằng rank của toán tử/toán hạng hiện tại.
+ Khi chạy hết toàn bộ vòng lặp nếu có braket != 0 thì trả về false.
+ Trả về true.

```
bool CheckExpression(string s)
{
int check = 0, bracket1 = 0, bracket2 = 0, bracket3 = 0;
for (int i = 0; i < s.length(); i++)
{
if (s[i] == '(')
bracket1++;
if (s[i] == ')')
bracket1--;
if (s[i] == '[')
{
bracket2++;
s[i] = '(';
}
if (s[i] == ']')
{
bracket2--;
s[i] = ')';
}
if (s[i] == '{')
{
bracket3++;
s[i] = '(';
```

```
}
if (s[i] == '}')
{
bracket3--;
s[i] = ')';
}
if (bracket1 < 0 || bracket2 < 0 || bracket3 < 0)
return false;
if (check == 1)
{
if (Rank(s[i]) == 1 && Rank(s[i - 1]) == 0)
return false;
if (Rank(s[i]) == 4)
return false;
}
if (check == 2 || check == 3)
{
if (Rank(s[i]) == 2 || Rank(s[i]) == 3)
return false;
}
if (check == 4)
{
if (Rank(s[i]) == 2 || Rank(s[i]) == 3)
return false;
}
if (check == 5)
{
if (Rank(s[i]) == 1)
return false;
}
if (Rank(s[i]) > 0)
check = Rank(s[i]);
}
if (bracket1 != 0 || bracket2 != 0 || bracket3 != 0)
return false;
return true;
}
```

Hàm string postfix(string s) trả về chuỗi là hậu tố từ phương trình trung tố s. Thuật toán đã trình bày ở 1.1

```
string postfix(string s)
{
string s_new = "";
stack<char> temp;
for (int i = 0; i < s.length(); i++)
{
	if (Rank(s[i]) == 0 && s_new[s_new.length() - 1] != ' ')
	{
		s_new += s[i];
		continue;
	}
	if (Rank(s[i]) == 1)
	{
		s_new += s[i];
		continue;
	}
```

```
        if (Rank(s[i]) == 2)
        {
                if (temp.size() > 0 && Rank(temp.top()) <= 3 && Rank(temp.top()) >= 2)
                {
                if (s_new[s_new.length() - 1] != ' ')
                        s_new += ' ';
                s_new += temp.top();
                temp.pop();
                }
                temp.push(s[i]);
                continue;
        }
        if (Rank(s[i]) == 3)
        {
                if (temp.size() > 0 && Rank(temp.top()) == 3)
                {
                if (s_new[s_new.length() - 1] != ' ')
                        s_new += ' ';
                s_new += temp.top();
                temp.pop();
                }
                temp.push(s[i]);
                continue;
        }
        if (Rank(s[i]) == 4)
                temp.push(s[i]);
        if (Rank(s[i]) == 5)
        {
                while (temp.top() != '(')
                {
                if (s_new[s_new.length() - 1] != ' ')
                        s_new += ' ';
                s_new += temp.top();
                temp.pop();
                }
                temp.pop();
        }
}
while (temp.size() > 0)
{
        if (s_new[s_new.length() - 1] != ' ')
                s_new += ' ';
        s_new += temp.top();
        temp.pop();
}
return s_new;
}
```

Hàm float calculation(string s) trả về giá trị của biểu thức từ chuỗi s là phép tính hậu tố. Ý tưởng:

- Tạo stack rỗng.
- Tạo vòng lặp với điều kiện dừng là đọc hết chuỗi s.
- Nếu gặp toán hạng thì đẩy vào stack.
- Nếu gặp toán tử thì lấy 2 toán hạng từ stack và tính toán sau đó đẩy giá trị tính được vào stack.

- Thực hiện đến hết chuỗi, trong stack sẽ còn lại 1 giá trị đó là kết quả của phép tính.

```
float calculation(string s)
{
stack<float> temp;
stringstream ss(s);
float f, float_temp1, float_temp2;
char c;
while (!ss.eof())
{
	ss >> c;
	if (c >= '0' && c <= '9')
	{
		ss.seekg(-1, ios::cur);
		ss >> f;
		temp.push(f);
		continue;
	}
	if (temp.size() >= 2)
	{
		float_temp2 = temp.top();
		temp.pop();
		float_temp1 = temp.top();
		temp.pop();
		if (c == '+')
		temp.push(float_temp1 + float_temp2);
		if (c == '-')
		temp.push(float_temp1 - float_temp2);
		if (c == '*')
		temp.push(float_temp1 * float_temp2);
		if (c == '/')
		temp.push(float_temp1 / float_temp2);
		if (c == '^')
		temp.push(pow(float_temp1, float_temp2));
	}
}
return temp.top();
}
```

Hàm main cho phép người dùng nhập đường dẫn đến file nhập - xuất và hiển thị ra màn hình: (c) giá trị tính toán của các biểu thức từ tệp đầu vào lấy 2 chữ số thập phân hoặc (t) phép tính ở dạng hậu tố. Nếu biểu thức lỗi cho kết quả là E.

```
int main()
{

    ifstream inPut;
    ofstream outPut;
    string s, s1, PostFix;
    string choice;
    int n;
    cout << "nhap file txt input: " << endl;
    cin >> s;
    inPut.open(s);
    cout << "nhap file txt output: " << endl;
```

```
    cin >> s1;
    outPut.open(s1);
    cout << "nhap so luong phep tinh: " << endl;
    cin >> n;
    cout << "chon hanh dong: " << endl;
    cout << "c: tinh toan " << endl;
    cout << "t: chuyen doi" << endl;
    cin >> choice;
    string* a = new string[n];
    float* cal = new float[n];
    if (choice == "c") {
        for (int i = 0; i < n; i++)
        {

            getline(inPut, a[i]);
            if (CheckExpression(a[i]) == true)
            {
                PostFix = postfix(a[i]);
                cal[i] = calculation(PostFix);
                outPut << setprecision(3) << cal[i] << endl;
            }
            else
            {
                outPut << "E" << endl;
            }
        }
    }
    else if (choice == "t")
        for (int i = 0; i < n; i++)
        {

            getline(inPut, a[i]);
            if (CheckExpression(a[i]) == true)
            {
                PostFix = postfix(a[i]);
                outPut << PostFix << endl;
            }
            else
            {
                outPut << "E" << endl;
            }
        }
    else
        cout << "nhap khong hop le " << endl;
}
```

TÀI LIỆU THAM KHẢO

https://yinyangit.wordpress.com/2011/01/26/algorithm-chuy%E1%BB%83n-bi%E1%BB%83u-th%E1%BB%A9c-trung-t%E1%BB%91-sang-ti%E1%BB%81n-t%E1%BB%91-va-h%E1%BA%ADu-t%E1%BB%91-b%E1%BA%B1ng-stack/

https://www.youtube.com/watch?v=MyCBpybscNo

https://www.youtube.com/watch?v=OPmN_YYQmro

https://www.stdio.vn/giai-thuat-lap-trinh/ung-dung-stack-bieu-thuc-tien-to-prefix-hspu1