PROJEKT ONP– DOKUMENTACJA

Zadaniem było zaimplementowanie algorytmu ONP, a mianowicie:

-konwersja wyrażenia z notacji tradycyjnej do ONP

-obliczanie wartości wyrażenia w ONP

Do implementacji ONP użyłem reprezentacji tablicowej stosu:

```
const int ROZMIAR = 100;  // definiuje rozmiar stosu

typedef int elementtype;
typedef int position;

class Stos
{
protected:
    int S[ROZMIAR];
    position Top;//szczyt stosu
public:
    Stos();
    void Push(elementtype x);
    void Pop();
    bool Empty();
    elementtype TopElem();
    void Makenull();
};

Stos::Stos()
{
    Top = -1;
}

void Stos::Makenull()
{
    Top = -1;
}
```

```
void Stos::Push(elementtype x)
{
    if (Top < ROZMIAR - 1)
    {
        Top++;
        S[Top] = x;
    }
}// PUSH

void Stos::Pop()
{
    if (Top >= 0) Top--;
}//POP

bool Stos::Empty()
{
    return (Top == -1);
}//Empty

elementtype Stos::TopElem()
{
    if (Top >= 0) return S[Top];
}
```

,gdzie definiuje rozmiar Stosu jako 100, a nastepnie implementuje operacje: MAKENULL, PUSH, POP, EMPTY, TopElem, i Konstruktor.

Implementacja algorytmu do obliczania ONP:

Mój algorytm do obliczania wyrażenia w ONP jest to funkcja, która ja ko argument przyjmuje stringa, tworze zmienne arg1,arg2, wynik do obliczen.Cały algorytm siedzi w pętli while(nie_koniec_danych) i przepatrujemy po każdym znaku z poczatkowego stringa. W tej pętli funkcja isspace()sprawdzam czy znak nie jest spacją, nastepnie robię if i sprawdzam czy znak nie jest cyfrą, jeśli tak to dodaje ten znak do zmiennej finalny, następnie funkcja atof() konwertuje stringa na double i funkcją Push wstawiam element na stos. Jeśli znak nie jest cyfrą to przechodzę do else i tam zależnie od operatora (+,-,*,/,^,~) wykonuje odpowiednie operacje, np. dla + funkcją TopElem() zwracam element z góry stosu, używam funkcji Pop() do usuniecia elementu z góry stosu i przypisuje do arg1, tak samo postepuje z arg2 i nastepnie do wyniku przypisuje arg1+arg2. Pozostałe operacje z operatorami wygladają anlogicznie.Na końcu odkładam wynik na stos funcją Push().

```
void ONP(string wyrazenie)
{
    Stos stosik=Stos();

    int i=0;
    double arg1, arg2;          // argumenty operacji
    string finalny = "";
    double  wynik;
    arg1 = wynik = arg2 = 0.0;

    while (i < wyrazenie.length())
    {
          while (isspace(wyrazenie[i])) ///sprawdza czy znak jest spacja
          {
              i++;
          }

          if (isdigit(wyrazenie[i])) ///sprawdza czy znak jest cyfra
          {
              while (isdigit(wyrazenie[i]))
              {
                  finalny += wyrazenie[i];
                  i++;
              }
              ///konwertuje stringa na double
             stosik.Push(atof(finalny.c_str()));
             finalny = "";
          }
          ///podajemy operator
         else {
         if (wyrazenie[i] == '~'){
              arg1 = stosik.TopElem();
              stosik.Pop();
              arg2=stosik.TopElem();
              stosik.Pop();
              wynik=(-arg1);
          }

          if (wyrazenie[i] == '+'){
              arg1 = stosik.TopElem();
              stosik.Pop();
              arg2 = stosik.TopElem();
              stosik.Pop();
              wynik = (arg1 + arg2);
          }
          if (wyrazenie[i] == '-'){
              arg1 = stosik.TopElem();
              stosik.Pop();
              arg2 = stosik.TopElem();
              stosik.Pop();
              wynik = (arg2 - arg1);
          }
          if (wyrazenie[i] == '*'){
              arg1 = stosik.TopElem();
              stosik.Pop();
              arg2 = stosik.TopElem();
              stosik.Pop();
              wynik = (arg1 * arg2);
          }
          if (wyrazenie[i] == '/'){
              arg1 = stosik.TopElem();
              stosik.Pop();
              arg2 = stosik.TopElem();
              stosik.Pop();
              wynik = (arg2 / arg1);
          }
          if (wyrazenie[i] == '^'){
              arg1 = stosik.TopElem();
              stosik.Pop();
              arg2 = stosik.TopElem();
              stosik.Pop();
              wynik = (pow(arg2,arg1));
          }

          i++;
          ///wloz wynik na stos
          stosik.Push(wynik);
          }
}
cout <<"Wynik naszego wyrazenia: "<<stosik.TopElem() << endl<<endl; // wypisujemy wynik ze szczytu stosu
```

Implementacja algorytmu do konwersji wyrażenia na wyrażenie zapisane w ONP:

Mój algorytm do konwersji wyrażenia do ONP to dwie funkcje, ta właściwa przyjmujaca stringa jako argument i drugaprzyjmujaca znak jako argument i zwracająca priorytet operatorów jako liczbe(najwiekszy priorytet(potegowanie) zwraca najwieksza liczbe).W funkcji konwertuj tworze stringa wyjscie które będzie moim wyjsciem.Implementacja algorytmu odbywa się w petli for i sprawdzamy tam po jednym znaku ze stringa. Na poczatku sprawdzam czy wyrazenie[i] jest znakiem, jeśli tak to do zmiennej wyjscie oddajemy ten znak. Jeśli znak =(to wstawiamy go na szczyt stosu funkcją PUSH. Jeśli znak =) to będziemy usuwać element ze stosu funkcją POP, jednak sprawdzamy czy stos nie jest pusty i czy element na szczycie stosu nie jest (, jeśli tak to do zmiennej wyjscie dodajemy element ze szczytu stosu i usuwamy go ze stosu funkcja POP. Nastepnie jest else odnośnie operatorów, jeśli znak jest operatorem to dodajemy go na stos funkcją PUSH, w else jest pelta while, gdzie sprawdzamy który z operatorow ma wiekszy priorytet i w tym while dodajemy do zmiennej wyjscie gorny element z stosu i usuwamy go funkcją POP. W zmiennej wyjsce mamy przekonwertowane wyrazenie.

```
int priorOperator(char c)
{
    if (c == '-' || c == '+')
        return 1;
    else if (c == '*' || c == '/')
        return 2;
    else if (c == '^')
        return 3;
    return 0;
}
void konwertujDoONP(string wyrazenie)
 {
     Stos stosik=Stos();
     wyrazenie='('+wyrazenie+')';
     string wyjscie;


    for (int i=0; i<wyrazenie.length(); i++)
    {
        ///sprawdzamy czy jest znakiem
        if(isdigit(wyrazenie[i])) wyjscie+=wyrazenie[i];
        else if(wyrazenie[i]=='(') stosik.Push(wyrazenie[i]);
        else if(wyrazenie[i] ==')')
        {
            while(!stosik.Empty() && stosik.TopElem()!='(')
            {
                wyjscie+=stosik.TopElem();
                stosik.Pop();
            }
            stosik.Pop();
```

```
        while(!stosik.Empty() && stosik.TopElem()!='(')
        {
            wyjscie+=stosik.TopElem();
            stosik.Pop();
        }
        stosik.Pop();
    }
    else
    {
        if(  !isdigit(stosik.TopElem()) )
        {
            while(priorOperator(wyrazenie[i])<=priorOperator(stosik.TopElem()))
            {
                wyjscie+=stosik.TopElem();
                stosik.Pop();
            }
            stosik.Push(wyrazenie[i]);
        }
    }
}
cout<<stosik.TopElem()<<endl;
cout<<"Wyjscie: "<<wyjscie<<endl<<endl;
```

Program główny to switch w nieskończonej petli while jeśli wybierzemy 1 to obliczamy wyrażenie w ONP , wyrażenie wpisujemy z klawiatury i musi być ono zakończone znakiem =.Jesli wybierzemy 2 to będziemy konewrtowac wyrażenie do ONP, wyrazenie wpisujemy z klawiatury i musi być ono zakończone znakiem =, które nie jest brane pod uwage.Jeśli wpiszemy q to nastapi wyjscie z programu.