**SPRAWOZDANIE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Przedmiot** | Algorytmy i struktury danych | **Zadanie** | 4\_01 |
| **Autor** | Kamil Borkowski | **Grupa** | WCY22IY1S1 |
| **Temat** | ONP | | |

1. Treść

Zaimplementuj algorytm ONP na stosach.

Metoda realizacji

Dane i zmienne dajemy na jeden stos, a operandy na drugi, następnie operandy dodajemy do stosu wyjścia według schematu.

* 1. Założenia / ograniczenia dotyczące danych:
     1. Dane wejściowe *(opis)*

Postać infiksowa – n – wczytywana z klawiatury

* + 1. Dane wyjściowe *(opis)*

Postać postfiksowa – wyświetlone na ekranie

1. Realizacja
   1. Algorytm *(lista kroków lub schemat blokowy)*

Deklaracja struktury listy wyjścia i listy operandów.

Uzupełnienie listy poprzez tworzenie nowych elementów listy

Usunięcie elementów nie będącymi liczbami pierwszymi

Wyświetlenie wszystkich elementów listy

* 1. Kod źródłowy

//================================================

//Zadanie 4\_01 jezyk C++

//ONP

//WCY21IY1S1 Borkowski Kamil

//================================================

#include <iostream>

#include <string>

using namespace std;

// Struktura reprezentująca stos operacji

struct ElementStosu

{

char wartosc;

ElementStosu\* nastepny;

};

// Struktura reprezentująca stos wyjścia (postfiks)

struct Wyjscie

{

char wartosc;

Wyjscie\* nastepny;

};

// Funkcja pomocnicza do dodawania elementu na stos operacji

void push(ElementStosu\*& head, char wartosc)

{

ElementStosu\* nowy\_element = (ElementStosu\*)malloc(sizeof(ElementStosu));

nowy\_element->wartosc = wartosc;

nowy\_element->nastepny = head;

head = nowy\_element;

}

// Funkcja pomocnicza do usuwania elementu ze stosu operacji

void pop(ElementStosu\*& head)

{

if (head == NULL)return;

ElementStosu\* temp = head;

head = head->nastepny;

free(temp);

}

// Funkcja do zwracania wartości najwyższego elementu na stosie operacji

char top(ElementStosu\* head)

{

if (head == NULL)return 0;

return head->wartosc;

}

// Funkcja do dodawania elementu na stos wyjścia

void push(Wyjscie\*& head, char wartosc)

{

Wyjscie\* nowy\_element = (Wyjscie\*)malloc(sizeof(Wyjscie));

nowy\_element->wartosc = wartosc;

nowy\_element->nastepny = head;

head = nowy\_element;

}

// Funkcja do usuwania elementu ze stosu wyjścia

void pop(Wyjscie\*& head)

{

if (head == NULL)return;

Wyjscie\* temp = head;

head = head->nastepny;

free(temp);

}

// Funkcja do zwracania wartości najwyższego elementu na stosie wyjścia

char top(Wyjscie\* head)

{

if (head == NULL)return 0;

return head->wartosc;

}

// Funkcja do wyświetlania zawartości stosu wyjscia

// Stos wyjscia przechowuje elementy w odwrotnej kolejnosci niz chciana, wiec przekladamy elementy do pomocniczego stosu

void wyswietlStos(Wyjscie\* head)

{

Wyjscie\* temp = head;

Wyjscie\* odwroconyStos = NULL;

// Odwracamy stos

while (temp != NULL)

{

push(odwroconyStos, top(temp));// Wkladamy wierzchni element stosu wyjscia do stosu odwroconego

pop(temp);

}

// Wyświetlamy odwrócony stos

while (odwroconyStos != NULL)

{

cout << top(odwroconyStos);

push(temp, top(odwroconyStos));// Wyswietlamy elementy stosu odwroconego

pop(odwroconyStos);

}

cout << endl;

}

int main()

{

cout << "Podaj wyrazenie matematyczne w postaci infiksowej: ";

string wyrazenie\_infiksowe;

getline(cin, wyrazenie\_infiksowe);

Wyjscie\* stos\_wyjscia = NULL; // Inicjalizacja stosu wyjścia

ElementStosu\* stos\_operacji = NULL;//Inicjalizacja stosu operacji

for(char znak:wyrazenie\_infiksowe)

{

if( (znak != '(') && (znak != ')') && (znak != '+') && (znak != '-') && (znak != '\*') && (znak != '/'))//Jezeli znak nie jest operandem uznajemy go za stala lub nazwe zmiennej

{

push(stos\_wyjscia, znak); //stale i nazwy zmiennych przesylamy na stos wyjscia

}

else if(znak == '(')

{

push(stos\_operacji, znak); // ( dopisujemy na stos operacji

}

else if(znak == ')')

{

while(top(stos\_operacji) != 0 && top(stos\_operacji) != '(') // przesylamy wszystkie operandy na stos wyjscia az do (

{

push(stos\_wyjscia, top(stos\_operacji));

pop(stos\_operacji);

}

pop(stos\_operacji); // usuwamy otwierający nawias ze stosu operacji

}

else // Jeśli znak jest operatorem dodajemy go do stosu wyjscia

{

while(top(stos\_operacji) != 0 && top(stos\_operacji) != '(' && ((znak == '+' || znak == '-') && (top(stos\_operacji) == '\*' || top(stos\_operacji) == '/')))

{

push(stos\_wyjscia, top(stos\_operacji));

pop(stos\_operacji);

}

push(stos\_operacji, znak);

}

}

while(top(stos\_operacji))// przenosimy pozostale operatory na stos wyjscia

{

push(stos\_wyjscia, top(stos\_operacji));

pop(stos\_operacji);

}

//wyświetlamy zawartość stosu wyjścia

cout << "Postac postfiksowa: ";

wyswietlStos(stos\_wyjscia);

return 0;

}

* 1. Dane wejściowe *(ekran uruchomienia programu)*

Obraz zawierający tekst, zrzut ekranu, czarne

Opis wygenerowany automatycznie

* 1. Dane wyjściowe *(ekran uruchomienia programu)*

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

1. Wnioski *(złożoność asymptotyczna algorytmu przy użyciu notacji* ***O lub innej i inne wnioski)***

Złożoność obliczeniowa programu wynosi: O(n), zależna od długości postaci infiksowej.