Recursivitate 2023

Formatori:
Tutor: Stiegelbauer Paul
Tutor: Iovanovici Alexandru
+1

Data de începere a cursului:
19.02.2024
Utilizatori înscriși
Calendar
Note

Recursivitate 2023

Recursivitatea este o tehnică de programare prin care o funcție se apelează pe sine pentru a rezolva o problemă mai mare. În mod general, aceasta presupune definirea unei funcții care se va apela repetat (pe sine insusi), până când se va ajunge la o soluție finală. Această soluție finală este de obicei rezultatul unei serii de operații efectuate de funcția recursivă.

Spre exemplu, să presupunem că vrem să calculăm factorialul unui număr n. Definim funcția factorial(n) astfel:

```
function factorial(n) {
  if (n == 1) {
    return 1;
  } else {
    return n * factorial(n - 1);
  }
}
```

În acest caz, funcția factorial se apelează pe sine până când se ajunge la valoarea 1, caz în care se întoarce rezultatul 1. În caz contrar, se calculează produsul dintre n și factorialul pentru n-1. Astfel, dacă vrem să calculăm factorialul lui 5, apelăm funcția astfel:

```
factorial(5);
```

Aceasta va apela funcția pentru valorile 4, 3, 2 și 1, iar rezultatul final va fi 120 (5 x 4 x 3 x 2 x 1).

Recursivitatea poate fi foarte utilă pentru rezolvarea problemelor care implică procesarea recursivă a datelor sau a structurilor de date, precum arbori sau liste. Cu toate acestea, trebuie utilizată cu grijă, deoarece poate duce la probleme de depășire a stivei (stack overflow) dacă nu este implementată corect.

Expresie prefix

O expresie aritmetică în notația prefix (sau notația poloneză prefixă) este o modalitate de a reprezenta o expresie matematică în care operatorii precedă operanzii. În notația prefix, operatorii apar înaintea operanzilor și nu este nevoie de paranteze pentru a specifica ordinea de evaluare a expresiilor.

Spre exemplu, în notația prefix, expresia aritmetică 3 + 4 * 5 este reprezentată astfel: + 3 * 4 5. În această notație, operatorul de adunare (+) apare înaintea operanzilor 3 și * 4 5, unde operatorul de înmulțire (*) apare înaintea operanzilor 4 și 5.

Pentru a evalua o expresie aritmetică în notația prefix, se parcurge expresia de la stânga la dreapta, se identifică operatorii și se aplică operația corespunzătoare asupra operanzilor.

Implementarea s-a facut la curs!

Expresie infix

Pentru a implementa un program C care evaluează o expresie aritmetică în notația infix, putem folosi o stivă pentru a păstra operatorii și operanzii în ordinea corectă de evaluare. Iată un exemplu de implementare (!!! este generat algoritmic: este necesar sa il analizati si validati):

```
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#define MAX_SIZE 100
/* Definim structura pentru stiva */
struct Stack {
    int top;
    int data[MAX_SIZE];
};
/* Functie de verificare a stivei goale */
int isEmpty(struct Stack *stack) {
    return (stack->top == -1);
}
/* Functie de verificare a stivei pline */
int isFull(struct Stack *stack) {
    return (stack->top == MAX_SIZE - 1);
}
/* Functie de adaugare a unui element in stiva */
void push(struct Stack *stack, int value) {
    if (isFull(stack)) {
        printf("Stiva este plina!\n");
        exit(EXIT_FAILURE);
    }
    stack->data[++stack->top] = value;
}
/* Functie de extragere a unui element din stiva */
int pop(struct Stack *stack) {
    if (isEmpty(stack)) {
        printf("Stiva este goala!\n");
        exit(EXIT_FAILURE);
    }
    return stack->data[stack->top--];
}
/* Functie de citire a unei expresii aritmetice de la tastatura */
void readExpression(char *expression) {
    printf("Introduceti o expresie aritmetica: ");
    fgets(expression, MAX_SIZE, stdin);
}
/* Functie de evaluare a expresiei aritmetice */
int evaluate(char *expression) {
    struct Stack *stack = malloc(sizeof(struct Stack));
    stack->top = -1;
    int i, len, num = 0, operand1, operand2, result;
    char ch, operator;
    len = strlen(expression);
    for (i = len - 1; i >= 0; i--) {
        ch = expression[i];
        /* Daca este operand, il adaugam in stiva */
        if (isdigit(ch)) {
            num = (num * 10) + (int)(ch - '0');
        } else if (ch == ' ') {
            if (num != 0) {
                push(stack, num);
                num = 0;
            }
        } else {
            /* Daca este operator, extragem operandele si aplicam operatia */
            operand1 = pop(stack);
            operand2 = pop(stack);
            operator = ch;
            switch (operator) {
                case '+':
                    result = operand1 + operand2;
                    break;
                case '-':
                    result = operand1 - operand2;
                    break;
                case '*':
                    result = operand1 * operand2;
                    break;
                case '/':
                    result = operand1 / operand2;
                    break;
```

```
                default:
                    printf("Operator nevalid: %c\n", operator);
                    exit(EXIT_FAILURE);
            }
            /* Adaugam rezultatul in stiva */
            push(stack, result);
        }
    }
    /* Returnam rezultatul final */
    return pop(stack);
}
/* Functia main */
int main() {
    char expression[MAX_SIZE];
    readExpression(expression);
    int result = evaluate(expression);
    printf("Rezultatul expresiei este: %d\n", result);
    return 0;
}
```

Bibliografie

https://staff.cs.upt.ro/~marius/curs/pc/old/notes2.pdf

Aplicatii propuse

1. https://staff.cs.upt.ro/~marius/curs/lp/tema3.html : Exercițiile 1 și 2

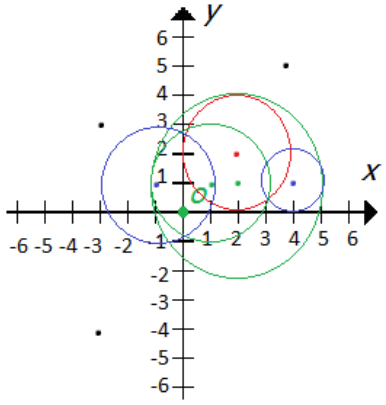
2. Se dau n bombe, numerotate de la 1 la n, pentru fiecare cunoscandu-se coordonatele (x,y) unde sunt plasate si raza de distrugere r. La explozia unei bombe se va distruge totul in interiorul si pe cercul de centru (x,y) si raza r, iar daca exista alte bombe in aceasta zona, acestea vor exploda la randul lor. Se da indicele k al primei bombe care explodeaza si se cere sa se calculeze cate bombe raman neexplodate.
Datele se citesc din fisierul bombe.in si rezultatele se vor afisa in fisierul bombe.out.
In fisierul bombe.in pe prima linie se afla numerele n si k, iar pe urmatoarele n linii coordonatele si razele de distrugere ale celor n bombe. n si k sunt numere naturale, coordonatele numere intregi, iar razele numere naturale.
Exemplu:

```
bombe.in
8 5
4 5 4
-3 -4 1
4 1 1
2 1 3
2 2 2
1 1 2
-1 1 2
-3 3 3
```

bombe.out

```
3
```

Explicatie: Prima explodeaza bomba rosie (a 5-a), ea declanseaza cele doua bombe verzi, iar fiecare dintre cele verzi declaseaza cate una albastra. Bombele negre raman neexplodate.



◄ Recursivitate 2022

Sari la...

Merge sort si altele ►

Sunteți conectat în calitate de [redacted]
S2-L-AC-CTIRO1-1C-TP

Meniul meu
- Profil
- Preferinte
- Calendar

ZOOM

Română (ro)
- English (en)
- Română (ro)

Rezumatul păstrării datelor
Politici utilizare site