DASTURLASHDA *RPN* DAN FOYDALANISH.

Esonov Sardorbek Abdurasul o‘g‘li
Mamatov Omadjon Musurmon o‘g‘li
Raimov Shomurod Keldiyor o‘g‘li
Mirzo Ulug’bek nomidagi O‘zMU Jizzax filiali talabalari.

Annotatsiya: RPNning o‘ziga xos xususiyati shundaki, barcha operandlar amal belgilaridan oldin joylashtiriladi. Bu esa ifodalarni baholash jarayonida samaradorlikni oshiradi va natijaviy hisoblashlarni tezlashtiradi. RPN dan dasturlashda foydalanish, operatorlarning to‘g'ri qo‘llanilishini ta'minlaydi va kalkulyatorlarda hisob-kitoblarni bajarishda qulaylik yaratadi. RPN qisqa yozuv va tez hisoblash imkonini berishi sababli, portativ va o‘rnatilgan hisoblash qurilmalari uchun muhim hisoblanadi.

Kalit so‘zlar: RPN (Teskari Polyak Yozuvi), matematik ifodalar, operatorlar, operandlar, hisoblash mashinalari, dasturlash tillari, infiks ifoda, precedens, amallar tartibi, qavslar, unar va binar amallar, natija, hisoblash jarayoni, kalkulyatorlar, dastur qismlari, xotira hajmi, portativ qurilmalar, amal belgilari, hisoblash tezligi, tajriba, operandlar soni, hisoblash natijasi.

RPN (Reverse Polish Notation – “teskari polyak yozuvi”) — bu matematik ifodalarni yozish usuli bo‘lib, unda operatorlar operandlardan keyin keladi. Bu usul, odatda, hisoblash mashinalarida va dasturlash tillarida qo‘llaniladi.

Teskari polyak yozuvi (RPN) avstraliyalik faylasuf va hisoblash bo‘yicha mutaxassis Charlz Hamblin tomonidan 1950-yillarning o‘rtalarida Polsha yozuviga asoslangan holda ishlab chiqilgan bo‘lib, u 1920-yilda polshalik matematik Yan Lukasievich tomonidan taklif qilingan. Hamblinning ishi 1957-yil iyun oyida bo‘lib o‘tgan konferentsiyada taqdim etilgan va 1957-1962-yillarda nashr etilgan.

Teskari Polsha yozuvining o‘ziga xos xususiyati shundaki, barcha argumentlar (yoki operandlar) operatsiya belgisidan oldin joylashtiriladi. Umuman olganda, kirish quyidagicha ko‘rinadi:

Operatsiyalar to‘plami yozuvi operandlar va operatsiya belgilari ketma-ketligidan iborat. Yozilganda ifodadagi operandlar bo‘sh joylar bilan ajratiladi.

Ifoda chapdan o‘ngga o‘qiladi. Ifodada amal belgisi uchrasa, uning oldidan uchragan oxirgi ikkita operandga ular yozilish tartibida tegishli amal bajariladi. Amaliyot natijasi uning operandlari ketma-ketligini va ifodadagi belgisini almashtiradi, shundan so‘ng ifoda xuddi shu qoida bo‘yicha qo‘shimcha baholanadi.

Ifodani baholash natijasi oxirgi baholangan operatsiya natijasidir.

Misol uchun, ifodani baholashni ko‘rib chiqing 7 2 3 * −(infiks belgisidagi ekvivalent ifoda: 7 − 2 * 3).

Operatsiyaning birinchi belgisi "*" dir, shuning uchun birinchi ko‘paytirish operatsiyasi 2 va 3 operandlarda amalga oshiriladi (ular belgidan oldin oxirgi keladi). Ifoda shaklga aylantiriladi 7 6 −(ko‘paytirish natijasi - 6 - "2 3 *" uchlik o‘rnini egallaydi).

Operatsiyaning ikkinchi belgisi - "-". 7 va 6 operandlarda ayirish amali bajariladi.

Hisoblash tugallandi. Oxirgi operatsiya natijasi 1, bu ifodani baholash natijasi.

Teskari polsha yozuvining unar, uchlik va boshqa operandlar soni bilan operatsiyalarga aniq kengayishi: ifodani baholashda bunday operatsiyalarning belgilaridan foydalanganda, operatsiya oxirgi duch kelgan operandlarning mos keladigan soniga qo‘llaniladi.

Teskari polyak yozuvining xususiyatlari quyidagilardan iborat:

Amaliyotlar tartibi ifodadagi ish belgilarining tartibi bilan bir ma’noda aniqlanadi, shuning uchun qavslardan foydalanish va operatsiyalarning ustuvorliklari va assotsiativligini kiritishning hojati yo‘q.

Infix yozuvidan farqli o‘laroq, unar va binar amallarni yozish uchun bir xil belgilardan foydalanish mumkin emas. Shunday qilib, infiks yozuvida ifoda 5 * (−3 + 8) unar amal (sonning ishorasini o‘zgartirish) belgisi sifatida minus belgisidan foydalanadi va (10 − 15) * 3ikkilik amalni (ayirish) ifodalash uchun xuddi shu belgidan foydalaniladi. Muayyan operatsiya belgining joylashgan joyi bilan

belgilanadi. Teskari polyak yozuvi bunga yo‘l qo‘ymaydi: belgi 5 3 − 8 + *(birinchi ifodaning shartli analogi) noto‘g‘ri talqin qilinadi, chunki 5 va 3 dan keyin "minus" ayirish emasligini aniqlash mumkin emas; natijada avval 5 − 3, keyin ni hisoblashga harakat qilinadi 2 + 8, shundan so‘ng ko‘paytirish operatsiyasi uchun operandlar etarli emasligi ma’lum bo‘ladi. Ushbu iborani yozish uchun siz uni qayta shakllantirishingiz kerak (masalan, ifoda o‘rniga − 3ifoda yozish orqali 0 − 3) yoki belgini o‘zgartirish operatsiyasi uchun alohida belgini kiritishingiz kerak, masalan, "±" 5 3 ± 8 + *:

Infix yozuvida bo‘lgani kabi, OPNda ham bir xil hisoblash bir necha xil usullarda yozilishi mumkin. Masalan, (10 − 15) * 3 arresterdagi ifoda sifatida yozilishi mumkin 10 15 − 3 *yoki u kabi yozilishi mumkin3 10 15 − *

Qavslar yo‘qligi sababli, teskari polyak yozuvi infiks belgisiga qaraganda qisqaroq. Shu sababli, kalkulyatorlarda hisob-kitoblarni amalga oshirishda operatorning ish tezligi oshadi (bosilgan tugmalar soni kamayadi), dasturlashtiriladigan qurilmalarda esa hisob-kitoblarni tavsiflovchi dastur qismlarining hajmi kamayadi. Ikkinchisi xotira hajmida qat’iy cheklovlarga ega bo‘lgan portativ va o‘rnatilgan hisoblash qurilmalari uchun muhim bo‘lishi mumkin.

RPNdan dasturlashda keng foydalanamiz. Misol tariqasida oddiy kalkulyator ilovasini yaratayotgan paytimizda operatorlarning to‘g‘ri qo‘llanilishi muammosiga duch kelamiz. Bunday muammolarda bizga RPNdan foydalanamiz.

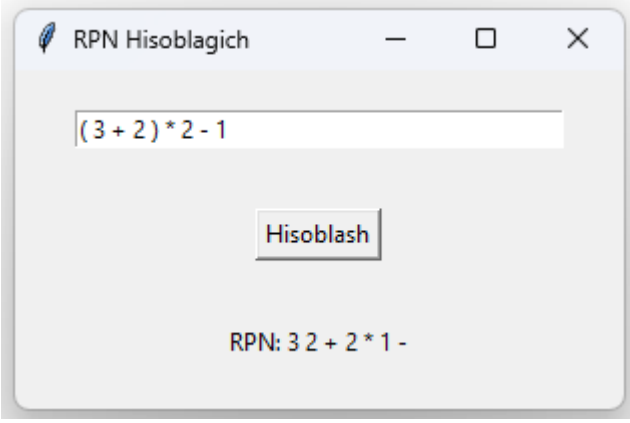
Quyidagi python kodi orqali RPNning qanday ishlashini dastur orqali ko‘rishimiz mumkin:

```
import tkinter as tk
def infix_to_rpn(expression):
    precedence = {'+': 1, '-': 1, '*': 2, '/': 2, '^': 3}
    output = []
    operators = []
    tokens = expression.split()
    for token in tokens:
```

```
            if token.isnumeric():  # Operand
                output.append(token)
            elif token in precedence:  # Operator
                while (operators and operators[-1] != '(' and
                       precedence[token] <= precedence[operators[-1]]):
                    output.append(operators.pop())
                operators.append(token)
            elif token == '(':
                operators.append(token)
            elif token == ')':
                while operators and operators[-1] != '(':
                    output.append(operators.pop())
                operators.pop()  # Remove '(' from stack
        while operators:
            output.append(operators.pop())
        return ' '.join(output)
    def calculate_rpn():
        expression = entry.get()
        try:
            rpn = infix_to_rpn(expression)
            result_label.config(text=f"RPN: {rpn}")
        except Exception as e:
            result_label.config(text="Xato: " + str(e))
    # Ilova yaratish
    root = tk.Tk()
    root.title("RPN Hisoblagich")
    # Kirish qutisi
    entry = tk.Entry(root, width=40)
    entry.pack(pady=20)
```

```
# Hisoblash tugmasi
calculate_button = tk.Button(root, text="Hisoblash",
command=calculate_rpn)
calculate_button.pack(pady=10)
# Natija ko‘rsatish
result_label = tk.Label(root, text="")
result_label.pack(pady=20)
# Ilovani ishga tushirish
root.mainloop()
```

Dastur natijasi quyidagicha:



1-rasm. RPN hisoblavchi dastur natiasi.

FOYDALANILGAN ADABIYOTLAR.

1. Aripova M.H, Qayumova M.H : Babamuxammedova M.Z. “Tizimli dasturiy ta’minot”, O‘quv qo‘llanma, Toshkent Axborot Texnologiyalari universiteti, 2016-yil.

2. Абылова Г. Ж. ОСОБЕННОСТИ СПЕЦИАЛИЗИРОВАННОЙ ПОДГОТОВКИ СТУДЕНТОВ НА ОСНОВЕ ДИДАКТИЧЕСКОЙ МОДЕЛИ В ПРОЕКЦИИ ЦИФРОВОГО ОБРАЗОВАНИЯ //Экономика и социум. – 2024. – №. 5-1 (120). – С. 1056-1061.

3. Abilova G., Shanazarov K., Shanazarova S. ANYLOGIC DASTURIY TA’MINOTNING IMKONIYATLARI VA AFZALLIKLARI //Академические исследования в современной науке. – 2023. – Т. 2. – №. 17. – С. 147-149.