acm.mipt.ru

олимпиады по программированию на Физтехе

Раздел «Язык Си» . StackCalculator :

Калькулятор выражений в обратной польской нотации

Рассмотрим запись арифметических выражений, в которых сначала следуют два операнда арифметической операции, а затем знак операции. Например:

Обратная польская нотация	Обычная нотация
2 3 +	2 + 3
(2 3 *) (4 5 *) +	(2 * 3) + (4 * 5)
2 3 4 5 6 * + - /	2 / (3 - (4 + (5 * 6)))

Заметьте, что скобки в обратной польской нотации не нужны. В частности, если во втором примере мы опустим скобки, выражение по прежнему будет интерпретироваться однозначно.

Транслятор этих выражений основан на стэке. Каждое следующее число помещается в стэк. Если встречается знак операции, то два числа из стэка извлекаются (a = pop(), b = pop()), для них вычисляется значение соответствующей бинарной арифметической операции и результат помещается в стек (push(a * b)).

НИже приведена постая реализация такого калькулятора на C. Заметим, что в ней нет никаких проверок ошибок ввода (если первый символ есть знак операции, она "умрёт"), а также стэк с его операциями pop и push не выделен в явную структуру. Роль стэка играет обычный массив (stack[256]) с указателем на вершину стэка (sp – stack pointer)

```
#include <stdio.h>
int
main()
{
    int stack[256];
    char buf[256];
    int sp = 0;
    printf("Sample:\n7 5 * 3 4 * + =\nResult = 47\n\nInput expression:\n")
    while(!feof(stdin))
    {
        if(scanf ("%s", buf) != 1 )
            break;
        switch(buf[0])
        {
            case '\0':
                break;
            case '=':
                printf("Result = %d\n", stack[--sp]);
                break;
            case '+':
               stack[sp-2] = stack[sp-2] + stack[sp-1];
               sp--;
               break;
            case '-':
               stack[sp-2] = stack[sp-2] - stack[sp-1];
               sp--;
               break;
            case '*':
               stack[sp-2] = stack[sp-1] * stack[sp-2];
               sp--;
               break;
            case '/':
               stack[sp-2] = stack[sp-1] / stack[sp-2];
               sp--;
```

Поиск
Поиск
Раздел «Язык Си»
Главная
Зачем учить С?
Определения
Инструменты:
Поиск
Изменения
Index
Статистика
Разделы
Информация
Алгоритмы
Язык Си
Язык Ruby
Язык Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C
Logon>>

```
                break;
            default:
                stack[sp++] = atoi(buf);
        }
    }
    printf("Result = %d\n",stack[sp-1]);
    return 0;
}
```

Реализация калькулятора на C++ с явным использованием стэка

```
#include <stdio.h>
#include <malloc.h>

class Stack {
    int *m_data;
    int m_size;
    int m_pt;
public:
    Stack(int size) {
        m_size = size;
        m_data = (int*)malloc(m_size * sizeof(int));
        m_pt = 0;
    };
    ~Stack() {
        free(m_data);
    };
    int pop(void) {
        if(m_pt)
            return m_data[--m_pt];
        else
            return 0;
    };
    void push(int a) {
        if(m_pt >= m_size-1) {
            m_size = 10 + 2 * m_size;
            m_data = (int*) realloc (m_data, m_size * sizeof(int));
        }
        m_data[m_pt++] = a;
    };
    int empty() {
        return (m_pt == 0);
    }
};

int
main() {
    class Stack s(3);
    int i;
    while(!feof(stdin)) {
        int c = getchar();
        int x;
        switch (c) {
            case EOF: break;
            case '\n':
            case ' ' : break;
            case '=' : printf("Result = %d\n", s.pop()); break;
            case 27  : goto RESULT;
            case '+' : s.push(s.pop() + s.pop()); break;
            case '-' : s.push(-s.pop() + s.pop()); break;
            case '*' : s.push(s.pop() * s.pop()); break;
            default:
                ungetc(c, stdin);
                if(scanf("%d", &x) != 1) {
                    fprintf(stderr, "Can't read integer\n");
                    return -1;
                } else {
                    s.push(x);
```

```
                }
                break;
            }
        }
RESULT:
    i = 0;
    while(!s.empty()){
        printf("Stack[%d] = %d\n", i,  s.pop());
        i++;
    }
    return 0;
}
```