Министерство образования Республики Беларусь
Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра программного обеспечения информационных технологий

Дисциплина: Метрология, стандартизация и сертификация (в информационных технологиях) (МСиСвИТ)

ОТЧЕТ
по лабораторной работе №1

Тема работы: Метрики размера программ

Выполнили
студенты: гр. 151003 Матошко И.В.
Барановский Р.А.

Проверил: Болтак С.В.

Минск, 2022

Анализируемый текст программы

```
#include <errno.h>
#include <stdio.h>
#include <stdlib.h>

#define STKDPTH 32

//Значения, возвращаемые функцией parse
#define VAL  0  //В стек занесено новое значение
#define ADD  1  //Сложение
#define SUB  2  //Вычитание
#define MUL  3  //Умножение
#define DIV  4  //Деление
#define SOF -1  //Переполнение стека
#define SUF -2  //В стеке недостаточно операндов
#define UNK -3  //Неопознанное значение

using namespace std;

//Глобальные переменные
int scount;
double stack[STKDPTH];

int parse(char* s)
{
    double tval = 0;
    char* endptr;

    //Распознаем знаки арифметических операций
    switch (*s) {
        case '-':
            //Если минус является первым и не последним
символом аргумента,
            //значит пользователь ввел отрицательное число
и опознавать его
            //как операцию вычитания не нужно
            if (*(s+1) != '\0') break;
            if (scount >= 2) {
                scount -= 1;
                return(SUB);
            }
            else return SUF;

        case '+':
            if (scount >= 2) {
                scount -= 1;
                return(ADD);
            }
```

```
            else return SUF;

        case 'x':
            if (scount >= 2) {
                scount -= 1;
                return(MUL);
            }
            else return SUF;

        case '/':
            if (scount >= 2) {
                scount -= 1;
                return(DIV);
            }
            else return SUF;
    }

    errno = 0;

    //Пытаемся сконвертировать строковый аргумент в число
    ptr = &endptr;
    tval = strtod(s, ptr);

    //Вернуть ошибку `неопознанный аргумент' в случае
неудачи
    if (errno != 0 || *endptr != '\0') return(UNK);

    //Проверяем, есть ли свободное место в стеке
    //и сохраняем в нем операнд, иначе возвращаем ошибку
переполнения
    if (scount < STKDPTH) stack[scount++] = tval;
    else return(SOF);

    return(VAL);
}

//Точка входа
int main(int argc, char** argv)
{
    //Организуем цикл для перебора аргументов командной
строки
    while (*++argv) {

        //Пытаемся распознать текущий аргумент как число
или
        //символ арифметической операции
     char* ptr = *argv;
        switch (ptr) {
            case VAL: continue;
```

```
            //Вычисляем
            case ADD:
                stack[scount - 1] += stack[scount];
                break;

            case SUB:
                stack[scount - 1] -= stack[scount];
                break;

            case MUL:
                stack[scount - 1] *= stack[scount];
                break;

            case DIV:
                if (stack[scount] != 0) {
                    stack[scount - 1] /= stack[scount];
                    break;
                } else {
                    cout << "Деление на ноль!\n";
                    return(1);
                }

            //Обработка ошибок
            case SUF:
                cout << "Недостаточно операндов!\n";
                return(1);

            case SOF:
                cout << "Переполнение стека!\n";
                return(1);

            case UNK:
                cout << "Неопознанный аргумент!\n";
                return(1);
        }
    }

    //Вывести результат
    int i;
    for (i = 0; i < scount; i++) cout << stack[i];

    return 0;
}



int doSomething() {
     input();
```

```
     int a = 8, x = 3;
     bool isGood = false;
     int* ptr = &a;
     while (a <= 4 && isOne(x)) {
          x += 7;
     } //WHats thatflm
     switch(x)
     {
     case 1:
          for (int i = 0, j = 1; i < x; ++i)
               a *= 8;
          cout << "Yes!";
          break;
     case 2:
          cout << "No!";
          break;
     case 3:
          a = x == 7 ? 0 : 1;
          cout << "This!";
          break;
     default:
          cout << "Two";
          *ptr = x;
     }
}

bool isOne(int a)
{
     return a == 1;
}

void input()
{
for(int j = 4; j <= 8^3; i = i + 1)
     cin >> b;
}

int partition(int* list, int start, int pivot)
{
     int i = start;
     while(i < pivot)
     {
          if(list[i] > list[pivot] && i == pivot-1)
          {
               int temp = list[i];
               list[i] = list[pivot];
               list[pivot] = temp;
               pivot--;
          }
```

```
          else if(list[i] > list[pivot])
          {
               int temp = list[pivot-1];
               list[pivot-1] = list[pivot];
               list[pivot] = temp;
               int temp = list[i];
               list[i] = list[pivot];
               list[pivot] = temp;
               pivot--;
          }

          else i++;
     }
     return pivot;
}

void quickSort(int* list, int start, int end)
{
     if(start < end)
     {
          int pivot = partition(list, start, end);
          int lb = pivot - 1, rb = pivot + 1;
          quickSort(list, start, lb);
          quickSort(list, rb, end);
     }
}

void mergeSort(int* list, int start, int end)
{
     int mid;
     if (start < end){

          mid=(start+end)/2;
          rMid = mid + 1;
          mergeSort(list, start, mid);
          mergeSort(list, rMid, end);
          merge(list,start,end,mid);
     }
}

void merge(int* list,int start, int end, int mid)
{
     int mergedList[8];
     int i, j, k;
     i = start;
     k = start;
     j = mid + 1;
```

```
        while (i <= mid && j <= end) {
            if (list[i] < list[j]) {
                mergedList[k] = list[i];
                k++;
                i++;
            }
            else {
                mergedList[k] = list[j];
                k++;
                j++;
            }
        }

        while (i <= mid) {
            mergedList[k] = list[i];
            k++;
            i++;
        }

        while (j <= end) {
            mergedList[k] = list[j];
            k++;
            j++;
        }

        for (i = start; i < k; i++) {
            list[i] = mergedList[i];
        }
    }
```

Таблица операторов

&&	3
:	16
*	9
[]	41
{}	32
||	1
;	112
()	13
=	41
switch()...case...default	3
if...else	13
+	6
!=	4
-	8
break	8
isOne()	1
>=	4
cin >>	1
-=	5
return	18
/=	1
&	2
,	19
merge()	1
strtod()	1
<	8
for()	4
++	14
>	2
while()	6
continue	1
+=	2
input()	1
*=	2
cout <<	9
<=	6
--	2
==	3
?...:	1
^	1
partition()	1
quickSort()	2
/	1
mergeSort()	2

Таблица операндов

SUF	5
scount	21
isGood	1
double	2
0	8
stack	12
4	2
"Переполнение стека!\n"	1
STKDPTH	2
'-'	1
3	3
"Недостаточно операндов!\n"	1
s	3
tval	3
ADD	2
endptr	3
VAL	2
1	25
'\0'	2
rb	2
2	6
rMid	2
SUB	2
'+'	1
temp	6
x	7
8	4
'x'	1
MUL	2
i	34
'/'	1
DIV	2
errno	2
ptr	6
argv	2
SOF	2
UNK	2
"Деление на ноль!\n"	1
"Неопознанный аргумент!\n"	1
a	6
false	1
7	2
j	12
"Yes!"	1
"No!"	1
"This!"	1
"Two"	1
mergedList	6

mid	8
b	1
start	11
pivot	18
list	29
k	11
end	9
lb	2

Словарь операторов (η1) = 44
Количество операндов (N1) = 431
Словарь операндов (η2) = 56
Количество операндов (N2) = 305
Словарь программы (η = η1 + η2) = 100
Длина программы (N = N1 +N2) = 736
Объем программы (V = $N\log_2\eta$) = 4889

Работа программы