МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

ТЕМА: РЕКУРСИЯ

Студент гр. 7382 _____ Глазунов С.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2018

Задание.

Вариант 11.

Написать программу, которая по заданному (см. предыдущее задание) константному_выражению вычисляет его значение либо сообщает о переполнении (превышении заданного значения) в процессе вычислений.
Например:2+2*2=6

Описание алгоритма.

Программа считывает строку, затем рекурсивно начинает складывать или вычитать относительно конца строчки,за исключением момента,когда встречается умножение. Умножение происходит тоже рекурсивно чтобы было возможно несколько умножений подряд. То есть получается рекурсия в рекурсии.

Описание функций и структур данных.

Функция bool is_number(std::string text).

Имеет 1 аргумент: *string text*.

string text – строка, содержащая изначальный текст.

Функция имеет тип *bool*, то есть возвращает *true*, либо *false*.

Функция проверяет число больше ли оно INT_MAX, если нет, то возвращает false, иначе *true*.

Функция void find_size_of_int(std::string text,int &i,int &size).

Имеет 3 аргумента: *string text*, *int* I,int size.

string text - строка, последовательность символов.

int i – сдвиг от начала текста.

Int size-размер числа.

Функция ничего не возвращает,но меняет i и size,т.к. они переданы по ссылке. Функция использует функцию isdigit,чтобы проверить каждый разряд -является ли это цифрой или нет.

Функция int multiply(std::string text,int &shift,int pad,int bonus_pad)

Имеет 4 аргумента:text,shift,pad,bonus_pad.

Text-изначальный текст.

Shift-сдвиг относительно начала текста.

Pad-глубина рекурсии.

bonus_pad-глубина второй рекурсии(умножения).

Функция превращает череду умножений в одно число,чтобы можно было складывать(вычитать) числа по прямому порядку.

Функция int recmath(std::string text,int shift,int pad)

Имеет 3 аргумента:text,shift,pad.

Text-изначальный текст.

Shift-сдвиг относительно начала текста.

Pad-глубина рекурсии.

Функция складывает или вычитает относительно конца строки. Это возможно только потому,что умножение представляется как одно число.

Функция *main*.

Функция не имеет аргументов.

Эта функция- главная функция программы.

В ней создается переменная *text* типа *string* и производится запись строки из потока ввода. Затем эта строка подается функции *recmath, которая и выдает результат*.

Тестирование.

Были написаны 14 тестов для данной программы, а также 2 скрипта для тестирования и компиляции программы.

Результаты тестирования на рис.1.

```
light5551@light5551-ThinkPad-E470:~/AiSD/7382/GlazunovSA/lab1$ ./runtest.sh
TEST#1 success output:25 [correct]
TEST#2 success output:-326 [correct]
TEST#3 success output:-1 [correct]
TEST#4 success output:4888330 [correct]
TEST#5 success output:16 [correct]
TEST#6 success output:-100000000 [correct]
TEST#7 success output:480 [correct]
TEST#8 success output:0 [correct]
TEST#9 success output:error [NOT correct]
TEST#10 success output:error [NOT correct]
TEST#11 success output:error [NOT correct]
TEST#12 success output:error [NOT correct]
TEST#13 success output:error [NOT correct]
TEST#14 success output:45 [ correct]
light5551@light5551-ThinkPad-E470:~/AiSD/7382/GlazunovSA/lab1$
```

Рисунок 1

Разберем тест 4.

На вход программа получает строку: “-1000+2445165*2+-1000”.

Программа сначала идет в самую глубь рекурсии,когда программа дошла до умножения 2445165*2 представляется как 4 890 330 и абстрактно строка превращается в «-1000+4890330+-1000».Дальше программа доходит до конца и начинает считать с конца, то есть строка абстрактно превращается в „-1000+4889770“.И так далее…. . В итоге получается окончательный ответ.

Выводы.

В результате работы были усвоены методы использования рекурсии, а также написана программа с использование метода рекурсии.

ИСХОДНЫЙ КОД.

```
#INCLUDE<IOSTREAM>
#INCLUDE<FSTREAM>
#INCLUDE<CLIMITS>//FOR INT_MAX AND INT_MIN
#INCLUDE<CSTDLIB>//FOR EXIT
#INCLUDE<CMATH>//FOR POW
INT STRTOINT(STD::STRING TEXT,INT SIZE);
//#DEFINE TEST          //FOR ILLUSTATION OF RECURSION
//#DEFINE SCRIPT_TEST   //FOR RUNTEST.SH
BOOL IS_NUMBER(STD::STRING TEXT)
{
        INT MAX_VALUE[10]={2,1,4,7,4,8,3,6,4,7};
        IF((TEXT[0]!='-'&&TEXT.SIZE()<10)||(TEXT[0]=='-'&&TEXT.SIZE()<11))
                RETURN TRUE;


        INT I=0;
        INT J=0;
        IF(TEXT[0]=='-')
                I=1;
        INT HIGHPOINT=10+I;
        FOR(;I<HIGHPOINT;I++)
        {
                IF(TEXT[I]>MAX_VALUE[J++])
                        RETURN FALSE;
        }


        RETURN TRUE;
}
VOID FIND_SIZE_OF_INT(STD::STRING TEXT,INT &I,INT &SIZE)
{


   IF(TEXT[I] == '-') {
      SIZE++;
      I++;
   }
   WHILE(ISDIGIT(TEXT[I]))
```

```
   {
      I++;
      SIZE++;
   }
}
INT MULTIPLY(STD::STRING TEXT,INT &SHIFT,INT PAD,INT BONUS_PAD)
{
   INT I=SHIFT;
   INT SIZE=0;
   INT TOTAL;
   STD::STRING CUR;

#IFDEF TEST                                    //THIS PART FOR ILLUSTRATION OF RECURSION
   FOR(INT K=0; K<PAD; K++)                          //
      STD::COUT<<"\T";                          //
                  FOR(INT K=0; K<BONUS_PAD; K++)                  //
                  STD::COUT<<" ";                           //
                        STD::COUT<<PAD<<"."<<BONUS_PAD<<".MULTIPLY"<<STD::ENDL;
//
#ENDIF

   FIND_SIZE_OF_INT(TEXT,I,SIZE);//IT'S A SEARCH OF NUMBER IN STRING
   CUR=TEXT.SUBSTR(SHIFT,SIZE);//TAKE A PART OF STRING FOR TRANSMUTATION IN
INT
   SHIFT=I;
   TOTAL=STRTOINT(CUR,SIZE);

   IF(TEXT[SHIFT]=='*')
   {
      SHIFT++;
      RETURN TOTAL*MULTIPLY(TEXT,SHIFT,PAD,++BONUS_PAD);
   }

   RETURN TOTAL;
}

INT RECMATH(STD::STRING TEXT,INT SHIFT,INT PAD) {//THE FIRST RECURSION
```

```
   IF(TEXT.SIZE()==SHIFT)//EXIT OF RECURSION
      RETURN 0;//

   INT SIZE=0,TOTAL,I=SHIFT;//TOTAL-IS ANSWER;
   STD::STRING CUR;

   FIND_SIZE_OF_INT(TEXT,I,SIZE);
   CUR=TEXT.SUBSTR(SHIFT,SIZE);//TAKE A PART OF STRING FOR TRANSMUTATION IN
INT
   SHIFT=I;
   TOTAL=STRTOINT(CUR,SIZE);

#IFDEF TEST
   FOR(INT K=0; K<PAD; K++)
      STD::COUT<<"\T";
   STD::COUT<<PAD<<".RECMATH"<<STD::ENDL;
#ENDIF
   IF(TEXT[SHIFT]=='+'){
      INT CURCHECK=RECMATH(TEXT,SHIFT+1,++PAD);

      IF(TOTAL>INT_MAX-ABS(CURCHECK))
{
#IFDEF SCRIPT_TEST
   FILE *F;
   F = FOPEN("OUTPUT_OF_LAB1.TXT","WT");
   FPRINTF(F, "ERROR" );
   FCLOSE(F);
#ELSE
   STD::COUT<<"ERROR"<<STD::ENDL;
      EXIT(0);
#ENDIF
}

   RETURN TOTAL+CURCHECK;
}

   IF(TEXT[SHIFT]=='*')//THE SECOND RECURSION(MULTIPLY)
   {
```

```
        SHIFT++;
        INT CURCHECK=MULTIPLY(TEXT,SHIFT,PAD,0);
        IF(CURCHECK!=0)
        IF(ABS(TOTAL)>INT_MAX/ABS(CURCHECK))

                {
                        #IFDEF SCRIPT_TEST
                FILE *F;
                F = FOPEN("OUTPUT_OF_LAB1.TXT","WT");
                FPRINTF(F, "ERROR" );
                FCLOSE(F);
                EXIT(0);
                        #ELSE
                STD::COUT<<"ERROR"<<STD::ENDL;
          EXIT(0);
                        #ENDIF
                }

       TOTAL*=CURCHECK;
       RETURN TOTAL+RECMATH(TEXT,SHIFT,++PAD);//
    }

    RETURN TOTAL;
}

INT STRTOINT(STD::STRING TEXT,INT SIZE) {       //THIS FUNCTION IS LIKE ATOI IN C
    INT TOTAL=0;
    INT I=0;
    INT CHECK=1;
    IF(TEXT[0]=='-') {
       I=1;
       CHECK=-1;
    }

    FOR(; I<SIZE; I++)
    {
       TOTAL+=((INT)(TEXT[I])-'0')*POW(10,SIZE-I-1);
    }
```

```
   IF(!IS_NUMBER(TEXT))//COMPARE SIZE OF NUMBER WITH NORMAL SITUATION
                    {
                    #IFDEF SCRIPT_TEST
                      FILE *F;
                    F = FOPEN("OUTPUT_OF_LAB1.TXT","WT");
                    FPRINTF(F, "ERROR" );
                    FCLOSE(F);
                    EXIT(0);
                    #ELSE
                    STD::COUT<<"ERROR"<<STD::ENDL;
          EXIT(0);
                    #ENDIF
                    }

   RETURN CHECK*TOTAL;
}

BOOL TEXTCORRECT(STD::STRING TEXT){                    //THIS FUNCTION CHECKS STDIN
   INT CHECK=0;
   IF(TEXT[TEXT.SIZE()-1]<'0'||TEXT[TEXT.SIZE()-1]>'9')// IF STDIN IS WRONG FOR
CALCULATOR
      RETURN FALSE;//CHECK LAST SYMBOL              //FUNC. SEND FALSE

   FOR(INT I=0;I<TEXT.SIZE();I++)
   {
      IF(ISALPHA(TEXT[I])||TEXT[I]=='.'||TEXT[I]==',')//CHECK SYMBOL IN STRING
         RETURN FALSE;
   }

   IF(TEXT[0]<'0'||TEXT[0]>'9')//CHECK THE FIRST SYMBOL
         IF(TEXT[0]!='-')
            RETURN FALSE;

   FOR(INT I=0;I<TEXT.SIZE();I++)// IF THERE IS A SITUATION LIKE "5+*3" -IT'S FALSE
   {
      IF(CHECK==2)
```

```
        RETURN FALSE;
      IF(TEXT[I]=='+'||TEXT[I]=='*')
      {
        CHECK++;
        CONTINUE;
      }
      CHECK=0;

   }

   FOR(INT I=0;I<TEXT.SIZE();I++)
        {
              IF(I!=0)
                    IF(TEXT[I]=='-'&&TEXT[I-1]!='+'&&TEXT[I-1]!='*')
                          RETURN FALSE;
        }
RETURN TRUE;
}

INT MAIN()
{
   STD::STRING TEXT;
   INT FIN;
#IFNDEF SCRIPT_TEST
   STD::COUT<<"HELLO,I AM CALCULATOR.ENTER..."<<STD::ENDL;
#ENDIF
   GETLINE(STD::CIN,TEXT);
   FOR(INT I=0;I<TEXT.SIZE();I++)
              IF(TEXT[I]==' ')
        {                          // "THIS IS AN EXAMPLE SENTENCE."
        TEXT.ERASE (I,1);
        I--;
        }


FOR(INT I=0; I<TEXT.SIZE(); I++)
      IF(TEXT[I]=='-')
      {
```

```
            IF(TEXT[I+1]=='-')
            {
               TEXT.ERASE (I,1);
               TEXT[I]='+';
               I--;

            }
         }

         FOR(INT I=0;I<TEXT.SIZE();I++)
      {
         IF(I&&TEXT[I]=='-')
            IF(TEXT[I-1]=='+'||TEXT[I-1]=='*')
            {
               CONTINUE;
            }
            ELSE IF(ISDIGIT(TEXT[I-1]))
            {
               TEXT.INSERT(I,"+");
            }
      }

         BOOL ISCORRECT=TEXTCORRECT(TEXT);
      IF(ISCORRECT)
      FIN=RECMATH(TEXT,0,0);
#IFDEF SCRIPT_TEST
      FILE *F;
      F = FOPEN("OUTPUT_OF_LAB1.TXT","WT");
      IF(ISCORRECT){
      FPRINTF(F, "%D",FIN);
      }
      ELSE
      {
         FPRINTF(F, "ERROR" );
      }
      FCLOSE(F);
#ELSE
      IF(!ISCORRECT){
```

```
        STD::COUT<<"ERROR"<<STD::ENDL;
        RETURN 0;
    }
    STD::COUT<<"RESULT="<<FIN<<STD::ENDL;
#ENDIF
    RETURN 0;
}
```