Calculator Documentation

Release 0.1

Muhammad Yaseen
Geetha Rajadurai
Ali Khozravi
Majid Zeraati
Mortaza Jalalvand

Mar 14, 2017

CONTENTS:


ICTP
403
M+ M- Mc ÷
7 8 9 x
4 5 6 −
1 2 3 +
0 . =
ICTP

CHAPTER

ONE

INTRODUCTION

1.1 Introduction

Calculator is a C++ progamm to solve basic mathematical expressions. It can solve basic mathematical expressions containing +, −, ⋆, /, ⋆⋆, and `()`. The development version of the package is available on Github.

1.1.1 Code for Main Function

```
#include <iostream>
#include "tokenizer.h"
#include "parser.h"

using std::cout;
using std::endl;
using namespace std;

int main () {
  Tokenizer tokenizer;
  std::vector<Token> res = tokenizer.split("1+2*3");
 // for (Token token : res) {
 //   std::cout << token.val << std::endl;
 // }
  Parser parse;
  double result = parse.expression(res);
  cout << result << endl;
  return 0;
}
```

1.1.2 Code for Tokenizer Function

```
#include "tokenizer.h"
#include <iostream>

std::vector<Token> Tokenizer::split (std::string str)
{
  std::vector<Token> result;
  for (int i=0; i<str.length(); ++i) {
    char c = str[i];
    if (c=='+')
    {
      std::string op;
      op += c;
      result.push_back(Token(PLUS,op));
    }
    else if (c=='-')
    {
      std::string op;
      op += c;
      result.push_back(Token(MINUS,op));
    }
    else if (c=='*')
    {
      std::string op;
      op += c;
      result.push_back(Token(STAR,op));
    }
    else if (c=='/')
    {
      std::string op;
      op += c;
      result.push_back(Token(SLASH,op));
    }
    else if (isblank(c)) continue;
    else if (isdigit(c))
    {
      std::string number;
      while(isdigit(str[i])) number+=str[i++];
      result.push_back(Token(NUMERIC,number));
      i--;
    }
    else
      std::cout<<"Unknown character"<<std::endl;
  }
  return result;
}
```

1.1.3 Code for Parser Function

```
#include "tokenizer.h"
#include "parser.h"


double Parser::factor(std::vector<Token> res)
{
```

```
  if (res[counter].kind == NUMERIC)
  {
    return stod (res[counter].val);
  }
//  else if (res[counter].kind == '(')
//   {
//     counter++; // '('
//     double result = expression(res);
//     counter++; // ')'
//     return result;
//   }
  else if (res[counter].kind == MINUS)
  {
    counter++;
    return -expression(res);
  }
  else if (res[counter].kind == PLUS)
  {
    counter++;
    return +expression(res);
  }
  else
    return 0; // error
}

double Parser::term(std::vector<Token> res)
{
  double result = factor(res);
  counter++;
  while (res[counter].kind == STAR || res[counter].kind == SLASH)
    if (res[counter].kind == STAR)
    {
      counter++;
      result *= factor(res);
    }
    else
    {
      counter++;
      result /= factor(res);
    }
  return result;
}

double Parser::expression(std::vector<Token> res)
{
  double result = term(res);
//   counter++;
  while (res[counter].kind == PLUS || res[counter].kind == MINUS)
    if (res[counter].kind == PLUS)
    {
      counter++;
      result += term(res);
    }
    else
    {
      counter++;
      result -= term(res);
    }
```

```
  return result;
}
```

Third test.

$$\frac{\sum_{t=0}^{N} f(t,k)}{N}$$

and inline math $\frac{\sum_{t=0}^{N} f(t,k)}{N}$.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search