Министерство образования и науки Российской федерации

Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Санкт-Петербургский Государственный Электротехнический Университет «ЛЭТИ» им. В. И. Ульянова (Ленина)» (СПбГЭТУ «ЛЭТИ»)

Факультет компьютерных технологий и информатики

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Кафедра вычислительной техники**

**Отчёт по лабораторной работе №\_\_**

по дисциплине «Программирование»

на тему

«Рекурсии С/C++»

Выполнил студент гр.5307 Самоуков Н.В.

Проверил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

"Выполнено" "\_\_\_\_"\_\_\_\_\_\_\_\_\_\_

Подпись преподавателя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Санкт-Петербург**

**2015 г.**

Оглавление

[Цель работы 3](#_Toc448478407)

[1.Задание 3](#_Toc448478408)

[2.Уточнение задания 3](#_Toc448478409)

[3.Контрольные примеры 3](#_Toc448478410)

[4.Описание переменных главной функции 4](#_Toc448478411)

[5.Краткое описание алгоритма 4](#_Toc448478412)

[6.Описание функций 5](#_Toc448478413)

[6.1 Функция is\_digit 5](#_Toc448478414)

[6.2 Функция in\_range 5](#_Toc448478415)

[6.3 Функция count 6](#_Toc448478416)

[6.4 Функция my\_atoi 7](#_Toc448478417)

[6.5 Функция len\_part 7](#_Toc448478418)

[6.6 Функция my\_getch 8](#_Toc448478419)

[6.7 Функция input\_yn 9](#_Toc448478420)

[6.8 Функция is\_cou 9](#_Toc448478421)

[6.9 Функция input\_string 10](#_Toc448478422)

[6.10 Функция countthis 11](#_Toc448478423)

[6.11 Функция is\_correct 11](#_Toc448478424)

[7.Текст программы 12](#_Toc448478425)

[8.Результат 18](#_Toc448478426)

[Вывод 18](#_Toc448478427)

# Цель работы

Получить практические навыки использования рекурсий в языке С/С++.

# 

# 1.Задание

Разработать алгоритм и написать программу, выполняющую многократно по желанию пользователя ввод математического примера, его обработку, вывод результата.

# 

# 2.Уточнение задания

Пример это строка, содержащая только цифры, '(',')', '\*','/','-','+'. Программа обрабатывает только целые числа от -9999 до 9999.

# 

# 3.Контрольные примеры

Контрольные примеры представлены в таблице 1. Контрольные примеры.

Таблица 1. Контрольные примеры

|  |  |  |
| --- | --- | --- |
| № | Входные данные | Ожидаемый результат |
| buf | i |
| 1 | "6+777-5" | 778 |
| 2 | "(6-56)\*55-3" | -2753 |
| 3 | "((55))\*56\*(-6)" | При вычислении возникла ошибка |
| 4 | "5\*(5-5)" | 0 |
| 5 | "0/0" | При вычислении возникла ошибка |
| 6 | "5\*5+6\*(-5)" | -5 |

# 4.Описание переменных главной функции

Переменные, используемые в главной функции, приведены в таблице 2 Переменные, используемые в главной функции.

Таблица 2. Переменные, используемые в главной функции

|  |  |  |
| --- | --- | --- |
| Имя | Тип | Назначение |
| buf | char\* | Указатель на первый символ выражения |
| i | int | Результат вычислений |
| error | bool | Флажок ошибки |
| exit | bool | Флажок выхода |

# 5.Краткое описание алгоритма

Шаг 1. Очистить экран.

Шаг 2. Вывести справку.

Шаг 3. Ввести выражение в buf

Шаг 4. Если выражение введено правильно, то Шаг 5 ,иначе Шаг 7.

Шаг 5. error = 0;

Вычисляем результат выражения и записываем его в i.

Шаг 6. Если !error , то вывести "Ответ:" + i, иначе

вывести "При вычислении возникла ошибка"

Шаг 8

Шаг 7. Вывести "Пример был введён неверно"

Шаг 8. Вывести "Продолжить работу?(y/n)"

Шаг 9. Вводим ответ в exit.

Шаг 10. Если exit , то конец программы , иначе Шаг 1.

# 6.Описание функций

# 6.1 Функция is\_digit

**Назначение функции:**

Это функция для проверки является ли числом символ.

**Прототип:**

bool is\_digit(unsigned char ch);

**Возвращаемое значение:**

Эта функция возвращает 1 если символ цифра, иначе 0.

**Пример вызова:**

is\_digit(ch)

**Описание переменных:**

Описание переменных в таблице 3. Переменные функции is\_digit.

Таблица 3. Переменные функции is\_digit

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ch | char | Символ |
| **Формальные параметры** | ch | unsigned char | Проверяемый символ |
| **Локальные переменные** | Локальные переменные отсутствуют. | | |

# 6.2 Функция in\_range

**Назначение функции:**

Это функция для проверки в пределе ли от -10000 до -10000 число.

**Прототип:**

bool in\_range(int i);

**Возвращаемое значение:**

Эта функция возвращает 1 если число в пределе, иначе 0.

**Пример вызова:**

in\_range(i)

**Описание переменных:**

Описание переменных в таблице 4. Переменные функции in\_range.

Таблица 4. Переменные функции in\_range

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | i | int | Результат расчётов |
| **Формальные параметры** | i | int | Проверяемое значение |
| **Локальные переменные** | Локальные переменные отсутствуют. | | |

# 6.3 Функция count

**Назначение функции:**

Это функция для расчёта выражения: "число" "знак" "число".

**Прототип:**

int count(int a, char d, int b, bool \*error);

**Возвращаемое значение:**

Эта функция возвращает результат расчётов.

**Пример вызова:**

i=count(4, '+', 6, &error);

**Описание переменных:**

Описание переменных в таблице 5. Переменные функции count.

Таблица 5. Переменные функции count

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | error | bool | Флажок ошибки при вычислении |
| **Формальные параметры** | a | int | Число 1 |
| d | char | Знак |
| b | int | Число 2 |
| error | bool \* | Указатель на флажок ошибки при вычислении |
| **Локальные переменные** | Локальные переменные отсутствуют. | | |

# 6.4 Функция my\_atoi

**Назначение функции:**

Это функция для перевода из строкового в целочисленный тип.

**Прототип:**

int my\_atoi(char \*in, int len);

**Возвращаемое значение:**

Эта функция возвращает число.

**Пример вызова:**

i=my\_atoi("555", 3);

**Описание переменных:**

Описание переменных в таблице 6. Переменные функции my\_atoi.

Таблица 6. Переменные функции my\_atoi

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | i | int | Результат расчётов |
| **Формальные параметры** | in | char \* | Указатель на первый символ строки |
| len | int | Длинна строки |
| **Локальные переменные** | с | int | Вспомогательная переменная |

# 6.5 Функция len\_part

**Назначение функции:**

Это функция для вычисления длины компонента.

**Прототип:**

int len\_part(char \*s);

**Возвращаемое значение:**

Эта функция возвращает длину строки.

**Пример вызова:**

len=len\_part(s);

**Описание переменных:**

Описание переменных в таблице 7. Переменные функции len\_part.

Таблица 7. Переменные функции len\_part

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | s | char \* | Вспомогательный указатель |
| **Формальные параметры** | s | char \* | Указатель на первый символ вводимой строки |
| **Локальные переменные** | i | int | Вспомогательная переменная |
| sko | int | Количество незакрытых скобок |

# 6.6 Функция my\_getch

**Назначение функции:**

Это функция для получения нажатой клавиши.

**Прототип:**

char my\_getch();

**Возвращаемое значение:**

Эта функция возвращает нажатую пользователем кнопку.

**Пример вызова:**

ch=my\_getch();

**Описание переменных:**

Описание переменных в таблице 8. Переменные функции my\_getch.

Таблица 8. Переменные функции my\_getch

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | ch | char | Символ нажатой клавиши |
| **Формальные параметры** | Формальные переменные отсутствуют | | |
| **Локальные переменные** | ch | char | Символ нажатой клавиши |

# 6.7 Функция input\_yn

**Назначение функции:**

Это функция для ввода ответа: да или нет.

**Прототип:**

bool input\_yn();

**Возвращаемое значение:**

Эта функция возвращает 0 или 1.

**Пример вызова:**

input\_yn()

**Описание переменных:**

Описание переменных в таблице 9. Переменные функции input\_yn

Таблица 9. Переменные функции input\_yn

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | i | int | Введённое число |
| **Формальные параметры** | Формальные переменные отсутствуют | | |
| **Локальные переменные** | sy | char | Введённый символ |
| ch | char | Нажатая клавиша |

# 6.8 Функция is\_cou

**Назначение функции:**

Это функция для проверки символа на возможность быть в примере.

**Прототип:**

bool is\_cou(char ch);

**Возвращаемое значение:**

Эта функция возвращает 1, если символ подходит, иначе 0.

**Пример вызова:**

is\_cou('4')

**Описание переменных:**

Описание переменных в таблице 10. Переменные функции is\_cou.

Таблица 10. Переменные функции is\_cou

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | Фактические параметры отсутствуют. | | |
| **Формальные параметры** | ch | char | Нажатая клавиша |
| **Локальные переменные** | Локальные переменные отсутствуют. | | |

# 6.9 Функция input\_string

**Назначение функции:**

Это функция для ввода строки.

**Прототип:**

char \*input\_string(int size);

**Возвращаемое значение:**

Эта функция возвращает указатель на первый символ введённой строки.

**Пример вызова:**

s=input\_string(30);

**Описание переменных:**

Описание переменных в таблице 11. Переменные функции input\_string

Таблица 11. Переменные функции input\_string

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | size | int | Максимальная длинна строки |
| **Формальные параметры** | s | char \* | Введённая строка |
| **Локальные переменные** | s | char \* | Указатель на первый символ исходной строки |
| length | int | Длина строки |
| ch | char | Введённый символ |

# 6.10 Функция countthis

**Назначение функции:**

Это функция для расчёта примера.

**Прототип:**

int countthis(char\* in, int len, bool \*error);

**Возвращаемое значение:**

Эта функция возвращает результат выражения.

**Пример вызова:**

i=countthis("555-4", 5, &error);

**Описание переменных:**

Описание переменных в таблице 12. Переменные функции countthis.

Таблица 12. Переменные функции countthis

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | error | bool | Флажок ошибки при вычислении |
| **Формальные параметры** | in | char\* | Указатель на первый символ строки с примером |
| len | int | Длинна примера |
| error | bool \* | Указатель на флажок ошибки при вычислении |
| **Локальные переменные** | i | int | Вспомогательная переменная |
| fri\_len | int | Длинна первого компонента |
| sec\_len | int | Длинна второго компонента |
| fri\_mno | int | Множитель первого компонента |

# 6.11 Функция is\_correct

**Назначение функции:**

Это функция для проверки правильности ввода.

**Прототип:**

bool is\_correct(char \*s);

**Возвращаемое значение:**

Эта функция возвращает 1, если пример введён правильно, иначе 0.

**Пример вызова:**

is\_correct("3646")

**Описание переменных:**

Описание переменных в таблице 13. Переменные функции is\_correct.

Таблица 13. Переменные функции is\_correct

|  |  |  |  |
| --- | --- | --- | --- |
| **Вид параметра** | **Имя**  **переменной** | **Тип переменной** | **Назначение переменной** |
| **Фактические параметры** | Фактические параметры отсутствуют | | |
| **Формальные параметры** | s | char \* | Указатель на первый символ примера |
| **Локальные переменные** | sko | int | Количество незакрытых скобок |
| last | char | Предыдущий символ |

# 7.Текст программы

#include <conio.h>

#include <malloc.h>

#include <stdio.h>

#include <windows.h>

#include <locale.h>

//Функция для проверки является ли числом символ

bool is\_digit(unsigned char ch)

{

return ch >= '0' && ch <= '9';

}

//Функция для проверки в пределе ли от -10000 до -10000 число

bool in\_range(int i)

{

return i > -10000 && i < 10000;

}

//Функция для расчёта примера: "число" "знак" "число"

int count(int a, char d, int b, bool \*error)

{

if (\*error == 1)

return 0;

switch (d)

{

case '-':

a = a - b;

break;

case '+':

a = a + b;

break;

case '\*':

a = a \* b;

break;

case '/':

if (b != 0)

{

\*error = 0;

return a / b;

}

else

{

\*error = 1;

return 0;

}

break;

}

\*error = !in\_range(a);

return a;

}

//Функция для перевода числа из char\* в int

int my\_atoi(char \*in, int len)

{

if (\*in != '-')

{

int c = 0;

while (len > 0)

{

len--;

c = c \* 10 + \*in - '0';

in++;

}

return c;

}

else

return -my\_atoi(in + 1, len - 1);

}

//Функция для вычисления длины компонента

int len\_part(char \*s)

{

int i;

if (\*s == '(')//если компонент в скобках

{

int sko;

sko = 1;

i = 1;

while (sko > 0)

{

s++, i++;

if (\*s == '(')

sko++;

if (\*s == ')')

sko--;

}

}

else//если компонент число

{

i = 0;

if (\*s == '-')

{

i++;

s++;

}

while (is\_digit(\*s))

s++, i++;

}

return i;

}

//Функция для получения символа.

char my\_getch()

{

char ch;//Код нажатой кнопки.

do

{

ch = \_getch();

if (ch == '\r')

ch = '\n';

else

if (ch == '\0' || ch == -32)

\_getch();

else

if (ch == '\t')

ch = ' ';

} while (ch >= -128 && ch <= -17 || ch == '\0' || ch == -32);

return ch;

}

//Функция для ввода ответа y/n

bool input\_yn()

{

char sy; //Введённый символ

char ch; //Нажатая клавиша

sy = 0;

do

{

ch = my\_getch();

switch (ch)

{

case '\b':

printf("\b \b");

sy = 0;

break;

case 'y':

case 'Y':

case 'n':

case 'N':

sy = ch;

printf("\b%c", sy);

break;

}

} while (ch != '\n' || sy == 0);

printf("\n");

return sy == 'y' || sy == 'Y';

}

//Функция для проверки символа на возможность быть в примере

bool is\_cou(char ch)

{

if (ch >= '0' && ch <= '9')

return 1;

switch (ch)

{

case '(':

return 1;

case ')':

return 1;

case '\*':

return 1;

case '/':

return 1;

case '+':

return 1;

case '-':

return 1;

default:

return 0;

}

}

//Функция для ввода строки

char \*input\_string(int size)

{

char \*s; //Указатель на первый символ исходной строки

int length = 1;//Длина строки

char ch; //Введённый символ

s = NULL;

do

{

ch = my\_getch();

switch (ch)

{

case 27:

break;

case '\b':

if (length > 1)

{

printf("\b \b");

length--;

s = (char \*)realloc(s, sizeof(char)\*length);

}

break;

case '\n':

if (length > 1)

{

while (s[length - 2] == ' ')

{

length--;

s = (char \*)realloc(s, sizeof(char)\*length);

}

s[length - 1] = 0;

}

break;

default:

if (length <= size && (ch != ' ' || length > 1) && is\_cou(ch))

{

s = (char \*)realloc(s, sizeof(char)\*(length + 1));

printf("%c", ch);

s[length - 1] = ch;

length++;

}

break;

}

} while (ch != '\n' || length == 1);

printf("\n");

return s;

}

//Функция для расчёта примера

int countthis(char\* in, int len, bool \*error)

{

if (\*error == 1)

return 0;

if (len <= 0)

return 0;

int i; //Локальные переменные

int fri\_len;//Длинна первого компонента

int sec\_len;//Длинна второго компонента

int fri\_mno;//Множитель первого компонента

if (\*in == '-')

{

in++;

fri\_mno = -1;

len--;

}

else

fri\_mno = 1;

fri\_len = len\_part(in);

if (fri\_len + (1 - fri\_mno) / 2 < len)//Больше одного компонента

{

if (\*(in + fri\_len) == '+')

return count(countthis(in, fri\_len, error)\*fri\_mno, \*(in + fri\_len), countthis(in +

fri\_len + 1, len - fri\_len - 1, error), error);

else

if (\*(in + fri\_len) == '-')

return count(countthis(in, fri\_len, error)\*fri\_mno, '+', countthis(in + fri\_len,

len - fri\_len, error), error);

else

{

sec\_len = len\_part(in + fri\_len + 1);

while ((fri\_len + sec\_len + 1 < len) && \*(in + fri\_len) != '+' && \*(in +

fri\_len) != '-')

{

fri\_len += 1 + sec\_len;

sec\_len = len\_part(in + fri\_len + 1);

};

if (\*(in + fri\_len) == '-')

return count(countthis(in, fri\_len, error)\*fri\_mno, '+', countthis(in +

fri\_len, len - fri\_len, error), error);

else

return count(countthis(in, fri\_len, error)\*fri\_mno, \*(in + fri\_len),

countthis(in + fri\_len + 1, len - fri\_len - 1, error), error);

}

}

else//1 компонент

if (\*in == '(')

return fri\_mno \* countthis(in + 1, len - 2, error);

else

{

\*error = !in\_range(my\_atoi(in, len));

return my\_atoi(in, len)\*fri\_mno;

}

}

//Функция для проверки правильности ввода примера

bool is\_correct(char \*s)

{

int sko; //Количество незакрытых скобок

char last;//Предыдущий символ

last = '-';

if (\*s == '-')

\*s++;

sko = 0;

while (\*s)

{

switch (\*s)

{

case '(':

sko++;

if (is\_digit(last) || last == ')')

return 0;

break;

case ')':

sko--;

if (!is\_digit(last) && last != ')')

return 0;

break;

case '\*':

case '/':

case '+':

if (!is\_digit(last) && last != ')')

return 0;

break;

case '-':

if (!is\_digit(last) && last != ')' && last != '(')

return 0;

break;

default:

if (last == ')')

return 0;

if (!is\_digit(\*s))

return 0;

break;

}

last = \*s;

s++;

}

return sko == 0 && last != '-';

}

int main()

{

setlocale(0, "RU");

char \*buf = NULL;//Указатель на первый символ введённой строки

int i; //Вспомогательная переменная

bool error; //Флажок ошибки

bool exit; //Флажок выхода

do

{

system("cls");

puts("Это программа-калькулятор");

puts("Она работает только с \* / + -");

puts("Умножение надо писать даже там, где оно подразумевается.");

puts("Программа работает только с целыми числами от -9999 до 9999");

puts("Вводите выражение (например 3+5\*(-1) )");

free(buf);

buf = input\_string(79);

if (is\_correct(buf))

{

error = 0;

i = countthis(buf, strlen(buf), &error);

if (!error)

printf("Ответ: %i\n", i);

else

puts("При вычислении возникла ошибка");

}

else

puts("Пример был введён неверно");

puts("Продолжить работу?(y/n)");

exit = !input\_yn();

} while (!exit);

free(buf);

}

# 8.Результат

При выполнении программы получены результаты, совпадающие с контрольными примерами, представленными в таблице 1. Контрольные примеры. Ошибки не обнаружены.

# Вывод

В результате выполнения лабораторной работы получены практические навыки работы с рекурсиями в языке C/C++.