**Отчёт по лабораторной работе №2.**

**Задание:**

Разработать стековую машину.

**Решение:**

1. Для стека данных будем использовать следующую модель шаблонного класса:

//шаблон класса Стек

template<class T>

class TStack{

public:

T \*data;

int Top;

int maxsize;

explicit TStack(int size = SIZESTACK):Top(0),maxsize(size),data(new T[size]){}

~TStack(){delete[] data;}

inline void push(T ch) {

if(Top<maxsize) {

data[Top++]=ch;

}

else error("Stack full");

}

inline T pop() {

if(Top!=0) {

Top--;

return data[Top];

}

else error("Stack empty");

return 0.0;

}

inline int size(){return Top;}

inline bool isempty() {return (Top==0);}

};

1. Класс объекта виртуальной машины (VM) состоит из стека данных, массива символов кода поступающей на вход машины программы, таблицы переменных, математических методов для выполнения арифметических операция с данными, системных методов для загрузки переменных в стек, выгрузки из стека, добавления новых переменных в таблицу символов.

//класс виртуальной машины

class VM

{

//хэш таблица символов(переменных)

static hash\_map<string,double> table;

public:

VM(int sizeStack=SIZESTACK):Code(nullptr),Data(sizeStack),NameVar(new char[SIZENAME]),it(0) {}

~VM(){if(Code) delete[] Code; delete[] NameVar;}

TStack<double> Data;

char \*Code;

char\* NameVar; //временная переменная для хранения текущего имени

int it; //текущая позиция в коде

//получить из потока имя и скопировать его во внутр. переменную

void getname()

//скопировать массив исходного кода

void setCode(char \*ptr,int size = SIZEPROG)

//добавить переменную в таблицу символов

static void add\_var(const string& name)

//установить значение переменной из таблицы символов

static void set\_var(const string& name,double value)

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*математические методы\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//вычисление корня

void sqrt()

//сумма

void add()

//разность

void sub()

//деление

void div()

//умножение

void mul()

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*конец математических методов\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//вывод на экран

void print()

//запуск виртуальной машины

void start()

};

1. Для заполнения массива, содержащего непосредственно код программы, использованы функции из учебного примера:

void put\_commd( char\*\* prg, char value)

{

\*\*prg = value;

(\*prg)++;

}

void put\_value(char\*\* prg, double value)

{

char\* ptr = (char\*)&value;

for (int i=0; i<sizeof(double); i++)

{

char cur = ptr[i];

put\_commd(prg, ptr[i]);

}

}

void put\_strng(char\*\* prg,const char\* string)

{

int len = strlen((string));

if (len >= SIZENAME-1) len = SIZENAME;

strncpy((\*prg),(char\*)(string), SIZENAME);

\*prg += len+1;

}

1. В главной функции производится заполнение массива с кодом поступающей на вход машины программы, инициализация виртуальной машины и ее непосредственный запуск:

int main(int argc, char\* argv[])

{

char Program[SIZEPROG]; /\* код программы\*/

char \*ptr = Program;

//добавить переменную

VM::add\_var("x");

VM::add\_var("y");

//выдаст ошибку, так как уже есть такая переменная

VM::add\_var("x");

//установить значение

VM::set\_var("pi",3.1415); //выдаст исключение, так как pi не определена

VM::add\_var("pi");

VM::set\_var("pi",3.1415); //теперь ok

// так создаются переменные в программе ВМ

put\_commd(&ptr, CNVAR); put\_strng(&ptr, "x1");

put\_commd(&ptr, CPUSH); put\_value(&ptr, 2.0);

put\_commd(&ptr, CSVAR); put\_strng(&ptr, "x1");

//арифметические операции, подсчет корня квадратного из (3\*3 + 4\*4)

put\_commd(&ptr, CPUSH); put\_value(&ptr, 3);

put\_commd(&ptr, CPUSH); put\_value(&ptr, 3);

put\_commd(&ptr, CMULT);

put\_commd(&ptr, CPUSH); put\_value(&ptr, 4);

put\_commd(&ptr, CPUSH); put\_value(&ptr, 4);

put\_commd(&ptr, CMULT);

put\_commd(&ptr, CADD);

put\_commd(&ptr, CSQRT);

//выводим результат, останавливаем машину

put\_commd(&ptr, COUT);

put\_commd(&ptr, CHALT);

VM MyVm;

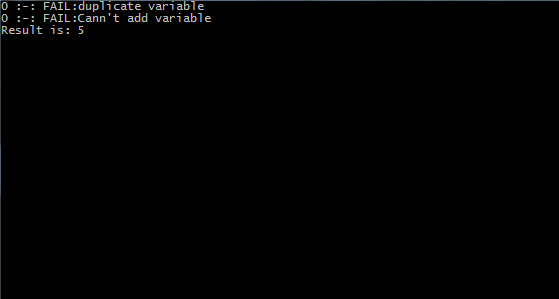
MyVm.setCode(Program);

MyVm.start();

return no\_of\_errors;

}

Пример работы программы (вычисление корня квадратного из выражения: 3\*3+4\*4):

****