**Отчёт по лабораторной работе №2.**

**Задание:**

Разработать стековую машину.

**Структура программы:**

1. Входные данные (байт-код) считываются из файла input.txt, расположенного в каталоге с исполняемым файлом. Формат файла представляет собой последовательность команд, если у команды есть аргумент, то значение аргумента идет следом за командой. В тестовом примере необходимо вычислить x=x+3\*5, где начальное значение x=2. Тогда input.txt принимает вид:

CNVAR 0

CPUSH 2

CSVAR 0

CPUSH 3

CPUSH 5

CMULT

CPULL 0

CADD

CSVAR 0

CPULL 0

COUT

CHALT

1. В главной функции производится чтение из файла input.txt, анализ полученных данных и формирование соответствующих команд виртуальной машине:

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

{

// код программы

char Source[SIZEPROG];

char \*pointer = Source;

string word;

int addr;

double value;

FILE\_input.open("input.txt",std::ios::in);

std::istream is(&FILE\_input);

//арифметические операции, (x1=x1+3\*5;)

while(is)

{

is >> word;

SWITCH (word) {

CASE("CSVAR"):

put\_commd(&pointer, CSVAR);

is>>addr;

put\_value(&pointer, addr);

break;

CASE("CNVAR"):

put\_commd(&pointer, CNVAR);

is>>addr;

put\_value(&pointer, addr);

break;

CASE("CPUSH"):

put\_commd(&pointer, CPUSH);

is>>value;

put\_value(&pointer, value);

break;

CASE("CPULL"):

put\_commd(&pointer, CPULL);

is>>addr;

put\_value(&pointer, addr);

break;

CASE("CMULT"):

put\_commd(&pointer, CMULT);

break;

CASE("CADD"):

put\_commd(&pointer, CADD);

break;

CASE("CSUB"):

put\_commd(&pointer, CSUB);

break;

CASE("CDIV"):

put\_commd(&pointer, CDIV);

break;

CASE("COUT"):

put\_commd(&pointer, COUT);

break;

CASE("CHALT"):

put\_commd(&pointer, CHALT);

break;

default:

break;

}

}

FILE\_input.close();

//поток вывода в файл output.txt

FILE\_output.open ("output.txt",std::ios::out);

std::ostream os(&FILE\_output);

//в умолчательном варианте конструктора используется поток cout

VM MyVm(&os);

MyVm.setCode(Source);

MyVm.start();

FILE\_output.close();

return no\_of\_errors;

}

1. Непосредственно сама виртуальная машина представляет из себя отдельный класс со стеком данных, массивом адресов переменных, поддерживающий методы арифметических операций с данными, а также работы со стеком адресов:

//класс виртуальной машины

class VM

{

pair<bool,double> variables[SIZEVART];

ostream\* output;

public:

VM(ostream\* out = &std::cout,int sizeStack=SIZESTACK):Code(nullptr),Data(sizeStack),it(0),output(out) {

for(int i=0;i<SIZEVART;i++)

//инициализация таблицы адресов

variables[i].first = false;

}

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

TStack<double> Data;

char \*Code;

int it; //текущая позиция в коде

//скопировать массив кода

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

if(Code) delete[] Code;

Code = new char[size];

memset(Code,0,size);

for(int i=0;i<size;i++)

{

Code[i] = \*(ptr+i);

}

}

//добавить переменную

void load(const int& addr)

{

if(addr>=SIZEVART) {error("addr out of range"); return;}

//адресс занят теперь

variables[addr].first = true;

variables[addr].second = 0.0;

it+=sizeof(int);

}

//установить значение переменной по адресу

void push(const int& addr)

{

if(addr>=SIZEVART) {error("addr out of range"); return;}

if(variables[addr].first)

variables[addr].second = Data.pop();

else

error("cann't find variable in table");

it+=sizeof(int);

}

//добавить в стек значений значение по адресу

void pull(const int& addr)

{

if(addr>=SIZEVART) {error("addr out of range"); return;}

if(variables[addr].first)

Data.push(variables[addr].second);

else

error("cann't find variable in table");

it+=sizeof(int);

}

//вычисление корня

void sqrt()

{

if(!Data.isempty())

{

double x = Data.pop();

Data.push(std::sqrt(x));

it++;

}

else error("cann't execute sqrt, stack is empty");

}

//сумма

void add()

{

if(Data.size()>1)

{

double x = Data.pop();

double y = Data.pop();

Data.push(x+y);

it++;

}

else error("cann't execute add, stack is empty");

}

//разность

void sub()

{

if(Data.size()>1)

{

double y = Data.pop();

double x = Data.pop();

Data.push(x-y);

it++;

}

else error("cann't execute add, stack is empty");

}

//деление

void div()

{

if(Data.size()>1)

{

double y = Data.pop();

double x = Data.pop();

if(y) Data.push(x/y);

else error("devide by zero");

it++;

}

else error("cann't execute add, stack is empty");

}

//умножение

void mul()

{

if(Data.size()>1)

{

double x = Data.pop();

double y = Data.pop();

Data.push(x\*y);

it++;

}

else error("cann't execute mul, stack is empty");

}

//вывод на экран

void print()

{

if(!Data.isempty()) \*output << "Result is: " << Data.pop() << std::endl;

else error("cann't res, stack is empty");

it++;

}

//запуск виртуальной машины

void start()

{

if(Code == nullptr) {

error("Not found code");

return;

}

while(1)

{

char cur = Code[it];

int curit = it;

switch (Code[it]) {

//методы для работы со стеком адресов

case CNVAR:

load(Code[++it]);

break;

case CSVAR:

push(Code[++it]);

break;

case CPULL:

pull(Code[++it]);

break;

//методы работы с данными

case CPUSH:

it++;

Data.push(\*((double\*)(Code+it)));

it+=sizeof(double);

break;

case CMULT:

mul();

break;

case CADD:

add();

break;

case CSUB:

sub();

break;

case CDIV:

div();

break;

case CSQRT:

sqrt();

break;

case COUT:

print();

break;

case CHALT:

return;

default:

error("Cann't understand command.");

break;

}

}

}

}

1. Результат работы виртуальной машины (а точнее оставшееся значение из стека данных) записывается в файл output.txt, например, в нашем тестовом примере это строка: «Result is: 17»