**Студент: Габдуллин Т.А.**

**Группа: М8О-206Б**

**Номер по списку: 20**

**Тема: Знакомство с языком МИКРОЛИСП.**

**Отображение программ из МИКРОЛИСПа в С++.**

**Лабораторная работа №1**

**Вариант: 23.06.1998**

**Распечатка файла even-odd.cpp**

**/\***

**(define(even-bits n)**

**(cond((= n 0)1)**

**((=(remainder n 2)0)**

**(even-bits (quotient n 2)))**

**(else(odd-bits(quotient n 2)))**

**))**

**(define(odd-bits n)**

**(cond((= n 0)0)**

**((=(remainder n 2)0)**

**(odd-bits (quotient n 2)))**

**(#t(even-bits(quotient n 2)))**

**))**

**(define(display-bin n)**

**(display(remainder n 2))**

**(if(= n 0)0 (display-bin (quotient n 2)))**

**)**

**(define(report-results n)**

**(display "Happy birthday to you!\n\t")**

**(display n)(display " (decimal)\n\t")**

**(display-bin n)(display "(reversed binary)\n")**

**(display "\teven?\t")(display (if(=(even-bits n)1) "yes" "no"))**

**(newline)**

**(display "\todd?\t")(display (if(=(odd-bits n)1) "yes" "no"))**

**(newline)**

**0**

**)**

**;\*\*\*\*\* Date of YOUR birthday \*\*\*\*\*\*\***

**(define dd 23)**

**(define mm 06)**

**(define yyyy 1998)**

**;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**(report-results (+ (\* dd 1000000)**

**(\* mm 10000)**

**yyyy))**

**\*/**

**#include "mlisp.h"**

**double even\_\_bits(double);**

**double odd\_\_bits(double);**

**double display\_\_bin(double);**

**double report\_\_results(double);**

**double dd = 23;**

**double mm = 06;**

**double yyyy = 1998;**

**double even\_\_bits(double n)**

**{**

**return (n == 0 ? 1**

**: remainder(n, 2) == 0 ?**

**even\_\_bits(quotient(n, 2))**

**: odd\_\_bits(quotient(n, 2)));**

**}**

**double display\_\_bin(double n) {**

**display(remainder(n, 2));**

**return(n == 0 ? 0 :**

**display\_\_bin(quotient(n, 2)));**

**}**

**double odd\_\_bits(double n)**

**{**

**return (n == 0 ? 0**

**: remainder(n,2) == 0 ?**

**odd\_\_bits (quotient(n,2))**

**: true ? even\_\_bits(quotient(n,2))**

**: \_infinity);**

**}**

**double report\_\_results(double n) {**

**display("Happy birthday to you!\n\t");**

**display(n);**

**display(" (decimal)\n\t");**

**display\_\_bin(n);**

**display("(reversed binary)\n");**

**display("\teven?\t");**

**display((even\_\_bits(n)) == 1 ? "yes" : "no");**

**newline();**

**display("\todd?\t");**

**display((odd\_\_bits(n)) == 1 ? "yes" : "no");**

**return 0;**

**}**

**/\* GTA \*/**

**int main(){**

**display(report\_\_results (dd\*1000000+**

**mm\*10000+**

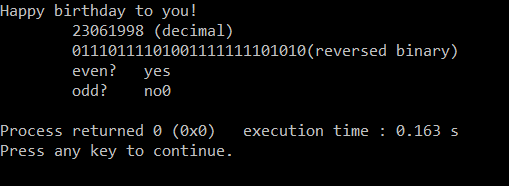
**yyyy));**

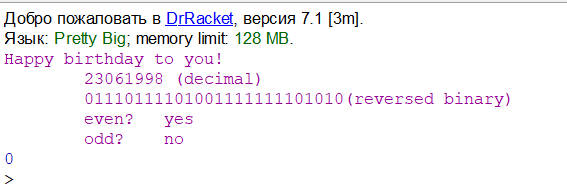
**newline();**

**return 0;**

**}**

**Скриншот запуска на C++**

****

**Скриншот запуска на Лиспе****\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Лабораторная работа №2**

**Вариант: 40**

**Распечатка файла half-interval.cpp**

**// half-interval.cpp 2018**

**/\***

**(define(fun z)**

**(set! z (- z (/ 40 41)(/ e)))**

**(- (\* 6 ( expt (atan (- z 2)) 4))**

**1)**

**)\*/**

**#include "mlisp.h"**

**double tolerance = 0.000001;**

**double fun(double z);**

**double average(double x, double y);**

**double close\_enough\_Q(double x, double y);**

**double \_\_gta\_\_try(double neg\_point, double pos\_point);**

**double half\_interval\_method(int a, int b);**

**double root(int a, int b);**

**double fun(double z) {**

**z=(z-40.0/41.0-1.0/ e);**

**return 6\*atan(z-2)\*atan(z-2)\*atan(z-2)\*atan(z-2) -1;**

**}**

**double average(double x, double y) {**

**return (x + y) / 2.0;**

**}**

**double close\_enough\_Q(double x, double y) {**

**return abs(x - y) < tolerance;**

**}**

**double \_\_gta\_\_try(double neg\_point, double pos\_point) {**

**{//let**

**double midpoint(average(neg\_point, pos\_point));**

**double test\_value(0);**

**display("+");**

**return close\_enough\_Q(neg\_point, pos\_point)**

**? midpoint**

**: true**

**? test\_value = fun(midpoint), (test\_value > 0)**

**? \_\_gta\_\_try(neg\_point, midpoint)**

**: (test\_value < 0)**

**? \_\_gta\_\_try(midpoint, pos\_point)**

**: midpoint**

**: \_infinity;**

**}**

**}**

**double half\_interval\_method(int a, int b) {**

**{//let**

**double a\_value(fun(a));**

**double b\_value(fun(b));**

**return (a\_value < 0 && b\_value > 0) ? \_\_gta\_\_try(a, b)**

**: (a\_value > 0 && b\_value < 0)**

**? \_\_gta\_\_try(b, a)**

**: b + 1;**

**}**

**}**

**double root(int a, int b) {**

**display("interval=\t[");**

**display(a);**

**display(" , ");**

**display(b);**

**display("]");**

**newline();**

**{//let**

**double temp = half\_interval\_method(a, b);**

**newline();**

**display("discrepancy=\t");**

**display(fun(temp));**

**newline();**

**display("root=\t\t");**

**display((temp - b - 1 == 0) ? "[bad]" : "[good]");**

**return temp;**

**}**

**}**

**//GTA\_ROOT**

**int main() {**

**display("GTA var 40");**

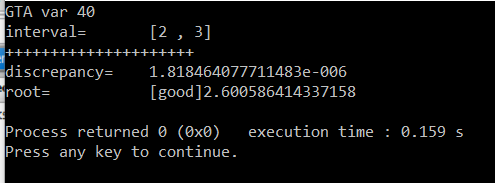
**newline();**

**display(root(2, 3));**

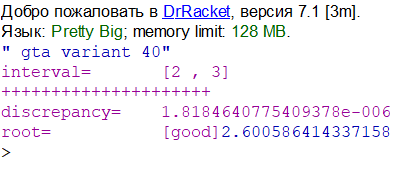
**newline();**

**}**

**Скриншот запуска на C++**



**Скриншот запуска на Лиспе**

****

**Лабораторная работа №3**

**Вариант: 20**

**Распечатка файла coin.ss**

**(define dd 23)**

**(define mm 06)**

**(define LARGEST-COIN 3)**

**(define (cc amount largest-coin)**

**(cond((or(= amount 0)(= largest-coin 1)) 1)**

**((not(and (> amount 0) (> largest-coin 0))) 0)**

**(else (+ (cc amount (next-coin largest-coin)) (cc (- amount largest-coin) largest-coin)))**

**))**

**(define (count-change amount)**

**(cc amount LARGEST-COIN)**

**)**

**(define (next-coin coin)**

**(cond((= coin 3) 2)**

**((= coin 2) 1)**

**(#t 0)**

**)**

**)**

**(define (GR-AMOUNT)**

**(+(\* 100 mm) dd))**

**(display " gta variant 20") (newline)**

**(display " 1-2-3") (newline)**

**(display "count-change for 100 \t= ")**

**(display (count-change 100)) (newline)**

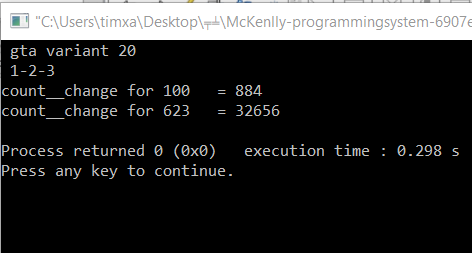
**(display "count-change for ")**

**(display (GR-AMOUNT))**

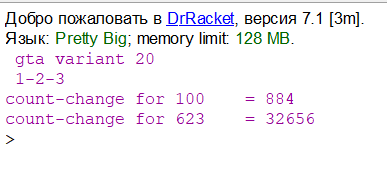
**(display " \t= ")**

**(display (count-change (GR-AMOUNT))) (newline)**

**Скриншот запуска на C++**



**Скриншот запуска на Лиспе**



**Вывод**

**В данных лабораторных работах я познакомился с языком MicroLisp. Это был ценный опыт, теперь могу писать программы лёгкой сложности на языке MicroLisp. Транслировать данный язык не так-то просто, как кажется.**

**В первой лабораторной работе требовалось преобразовать дату своего рождения в двоичное число и определить чётное ли количество единиц в нем с помощью готовой программы на языке MicroLisp.**

**Вторая лабораторная понравилась мне больше. Требовалось реализовать метод половинного деления для функции указанной в варианте с интервалом, указывающем на корень. Сложности возникли,но я их быстро исправил. Тем более метод половинного деления был написан на языке MicroLisp. Все что требовалось, это интерпретировать готовую программу в язык C++.**

**И наконец, в третьей лабораторной работе требовалось реализовать программу для вычисления количества способов размена монет с определенным номиналом(выставляется в варианте). А вот тут уже полученные знания о языке MicroLisp требовалось использовать на практике, чтобы произвести декомпиляцию с языка C++.**