# 2 Создание программной системы с элементарным интерфейсом консольного режима на С++

***Задание:***

Выполнить структурную декомпозицию, разработать структурную схему, содержащую не менее 3 подпрограмм, и алгоритмы этих подпрограмм. Реализовать на С++ в консольном режиме. Предусмотреть примитивный интерфейс типа меню, позволяющий выбирать нужную подпрограмму.

Разработать программу, которая выполняет простейшее исследование функций одной переменной. Реализовать следующие операции: ввод функции, преобразование ее во внутреннее представление (дерево), определение экстремумов функции на заданном отрезке, а также вывод результатов на экран.

2.1 Структурная декомпозиция программы

При создании программы процесс разработки разделился на следующие части:

определение наименьшего и наибольшего значений функции на отрезке;

определение значения функции в точке;

создание дерева;

удаление дерева.

2.2 Разработка схем алгоритмов программы

Разрабатываемые схемы алгоритмов должны обеспечить создание исходной программы (рисунки 1-5).

Рисунок 1 «Схема алгоритма основной программы»

Рисунок 2 «Схема алгоритма функции find\_min\_max»

Рисунок 3 «Схема алгоритма функции calculate»

Рисунок 4 «Схема алгоритма функции make\_tree»



Рисунок 5 «Схема алгоритма функции clear\_tree»

2.3 Разработка кода программы

Уточнение базовой структуры процедур и функций, используемых во время выполнения программы, позволило разработать код программы.

Ниже приведен код всей программы.

*#define \_USE\_MATH\_DEFINES*

*#include <iostream>*

*#include <sstream>*

*#include <cmath>*

Продолжение кода программы

*using namespace std;*

*enum Type { PLUS, MIN, MUL, DIV, SQU, SIN, COS, TG, CTG, LN, LG, VAL };*

*struct Tree*

*{*

*Type type;*

*Tree\* left, \* right;*

*string data;*

*};*

*void find\_min\_max(Tree\* tree, string s);*

*double calculate(Tree\* tree, double val);*

*void make\_tree(string str, Tree\* node, size\_t len);*

*void clear\_tree(Tree\* tree);*

*int main()*

*{*

*setlocale(LC\_ALL, "Rus");*

*string s("None");*

*Tree\* tree = nullptr;*

*while (true)*

*{*

*cout << "Выберите команду: " << endl;*

*cout << "0 - выход" << endl;*

*cout << "1 - ввод функции" << endl;*

*cout << "2 - нахождение экстремумов функции на отрезке" << endl;*

*cout << "3 - вывод текущей функции" << endl;*

*cout << "\nВаш выбор: ";*

*char ans;*

*cin >> ans;*

*cout << endl;*

*cin.ignore(2, '\n');*

*switch (ans)*

*{*

*case '0':*

*clear\_tree(tree);*

*return 0;*

*break;*

*case '1':*

*clear\_tree(tree);*

*cout << "Введите функцию y = ";*

*getline(cin, s);*

*tree = new Tree;*

*make\_tree(s, tree, s.length());*

*break;*

*case '2':*

*if (tree != nullptr)*

*find\_min\_max(tree, s);*

*else*

*cout << "Пожалуйста, введите функцию" << endl;*

*break;*

*case '3':*

*if (s != "None")*

*cout << "Текущая функция y = " << s << endl;*

Продолжение кода программы

*else*

*cout << "Нет введенной функции" << endl;*

*break;*

*default:*

*cout << "Команда " << ans << " неизвестна" << endl;*

*}*

*cout << endl;*

*system("pause");*

*system("cls");*

*}*

*return 0;*

*}*

*void find\_min\_max(Tree\* tree, string s)*

*{*

*double a, b, n = 1e-6;*

*string a\_string, b\_string;*

*bool key = false;*

*cout << "Введите концы отрезка [a; b]: ";*

*cin >> a\_string >> b\_string;*

*for (int i = 0; i < a\_string.length(); i++)*

*{*

*if (!isdigit(a\_string[i]))*

*{*

*key = true;*

*break;*

*}*

*}*

*for (int i = 0; i < b\_string.length(); i++)*

*{*

*if (!isdigit(b\_string[i]))*

*{*

*key = true;*

*break;*

*}*

*}*

*if (key)*

*{*

*cerr << "Введены неверные значения" << endl;*

*return;*

*}*

*a = atoi(a\_string.c\_str());*

*b = atoi(b\_string.c\_str());*

*double h = (b - a) \* 1.0 / n, min, max, min\_x = a, max\_x = a;*

*min = max = calculate(tree, a);*

*for (double x = a; x <= b; x += n)*

*{*

*double c = calculate(tree, x);*

*if (c > max) { max = c; max\_x = x; }*

*if (c < min) { min = c; min\_x = x; }*

*}*

*double c = calculate(tree, b);*

Продолжение кода программы

*if (c > max) { max = c; max\_x = b; }*

*if (c < min) { min = c; min\_x = b; }*

*cout << "\nФункция y = " << s << endl;*

*cout << "На отрезке [" << a << "; " << b << "]: " << endl << "min = y(" << min\_x << ") = " << min << "; max = y(" << max\_x << ") = " << max << endl;*

*}*

*double calculate(Tree\* tree, double val)*

*{*

*double res = 0;*

*switch (tree->type)*

*{*

*case PLUS:*

*res = calculate(tree->left, val) + calculate(tree->right, val);*

*break;*

*case MIN:*

*res = calculate(tree->left, val) - calculate(tree->right, val);*

*break;*

*case MUL:*

*res = calculate(tree->left, val) \* calculate(tree->right, val);*

*break;*

*case DIV:*

*if (calculate(tree->right, val) != 0)*

*res = calculate(tree->left, val) / calculate(tree->right, val);*

*else*

*{*

*cerr << "Деление на 0" << endl;*

*exit(1);*

*}*

*break;*

*case SQU:*

*res = pow(calculate(tree->left, val), calculate(tree->right, val));*

*break;*

*case SIN:*

*res = sin(calculate(tree->left, val));*

*break;*

*case COS:*

*res = cos(calculate(tree->left, val));*

*break;*

*case TG:*

*res = tan(calculate(tree->left, val));*

*break;*

*case CTG:*

*res = 1.0 / (tan(calculate(tree->left, val)));*

*break;*

*case LN:*

*if (calculate(tree->left, val) > 0)*

*res = log(calculate(tree->left, val));*

*else*

*{*

*cerr << "Аргумент ln должен быть > 0" << endl;*

*exit(1);*

Продолжение кода программы

*}*

*break;*

*case LG:*

*if (calculate(tree->left, val) > 0)*

*res = log10(calculate(tree->left, val));*

*else*

*{*

*cerr << "Аргумент lg должен быть > 0" << endl;*

*exit(1);*

*}*

*break;*

*case VAL:*

*if (tree->data == "x")*

*res = val;*

*else if (tree->data == "pi")*

*res = M\_PI;*

*else if (tree->data == "e")*

*res = M\_E;*

*else*

*res = atoi(tree->data.c\_str());*

*break;*

*default:*

*cerr << "Невозможно посчитать значение в функции в точке " << val << endl;*

*exit(1);*

*}*

*return res;*

*}*

*void make\_tree(string s, Tree\* tree, size\_t len)*

*{*

*unsigned i, c;*

*for (unsigned t = PLUS; t < VAL; ++t)*

*for (i = 0, c = 0; i < s.size(); ++i)*

*{*

*if ('(' == s[i])*

*++c;*

*else if (')' == s[i])*

*--c;*

*else if ((t == PLUS ? '+' : t == MIN ? '-' : t == DIV ? '/' : '\*') == s[i] && !c)*

*{*

*tree->left = new Tree;*

*tree->right = new Tree;*

*tree->type = static\_cast<Type>(t);*

*make\_tree(s.substr(0, i), tree->left, len);*

*make\_tree(s.substr(i + 1, s.size() - i), tree->right, len);*

*return;*

*}*

*else if (t == SQU && s[i] == '^' && !c)*

*{*

*unsigned j, c1;*

Продолжение кода программы

*tree->left = new Tree;*

*tree->right = new Tree;*

*tree->type = static\_cast<Type>(t);*

*make\_tree(s.substr(0, i), tree->left, len);*

*for (j = i + 1, c1 = 0; j < s.size() && (c1 || isdigit(s[j]) || s[j] == '('); ++j)*

*{*

*if ('(' == s[j])*

*++c1;*

*else if (')' == s[j])*

*--c1;*

*}*

*make\_tree(s.substr(i + 1, j - i - 1), tree->right, len);*

*return;*

*}*

*else if (s.size() - i >= 3)*

*{*

*if (((t == SIN ? "sin" : t == COS ? "cos" : "...") == s.substr(i, 3) ||*

*(s.substr(i, 3) == "ctg" && t == CTG) || (s.substr(i, 2) == "lg" && t == LG)*

*|| (t == TG ? "tg" : t == LN ? "ln" : "..") == s.substr(i, 2)) && !c)*

*{*

*tree->left = new Tree;*

*tree->right = nullptr;*

*tree->type = static\_cast<Type>(t);*

*if (t == SIN || t == COS || t == CTG)*

*{*

*unsigned j, c1;*

*for (j = i + 3, c1 = 0; j < s.size() && (c1 || s[j] == '('); ++j)*

*{*

*if ('(' == s[j])*

*++c1;*

*else if (')' == s[j])*

*--c1;*

*}*

*make\_tree(s.substr(i + 3, j - i - 1), tree->left, len);*

*}*

*else*

*{*

*unsigned j, c1;*

*for (j = i + 2, c1 = 1; j < s.size() && c1; ++j)*

*{*

*if ('(' == s[j])*

*++c1;*

*else if (')' == s[j])*

*--c1;*

*}*

Продолжение кода программы

*make\_tree(s.substr(i + 2, j - i - 1), tree->left, len);*

*}*

*return;*

*}*

*}*

*}*

*if (s[0] == '(' && s[s.size() - 1] == ')')*

*{*

*make\_tree(s.substr(1, s.size() - 2), tree, len);*

*return;*

*}*

*if (s.length() == len && s != "x")*

*{*

*cerr << "\nНеизвестная функция y = " << s << endl;*

*exit(1);*

*}*

*tree->type = VAL;*

*stringstream istr(stringstream::in | stringstream::out);*

*istr << s;*

*istr >> tree->data;*

*tree->left = nullptr;*

*tree->right = nullptr;*

*}*

*void clear\_tree(Tree\* tree)*

*{*

*if (!tree) return;*

*if (tree->left) clear\_tree(tree->left);*

*if (tree->right) clear\_tree(tree->right);*

*delete tree;*

*}*

2.4 Тестирование программы

Разработанное консольное приложение позволяет выполнять функции, предусмотренные заданием (рисунки 6-9) .

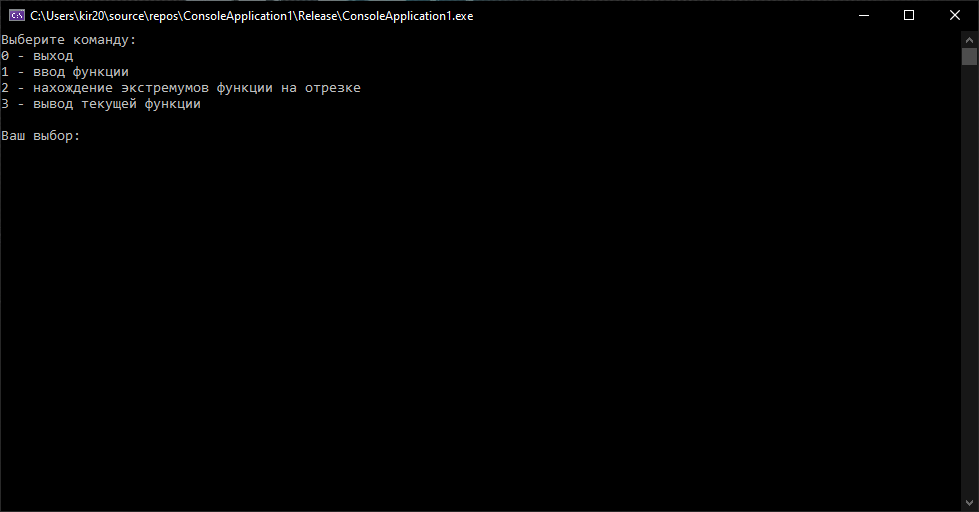


Рисунок 6 «Меню»

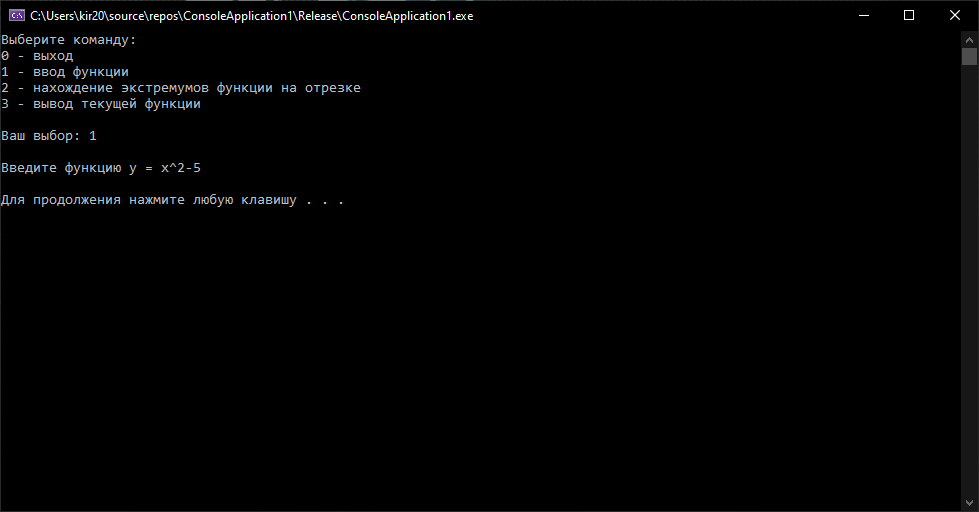


Рисунок 7 «Добавление функции»

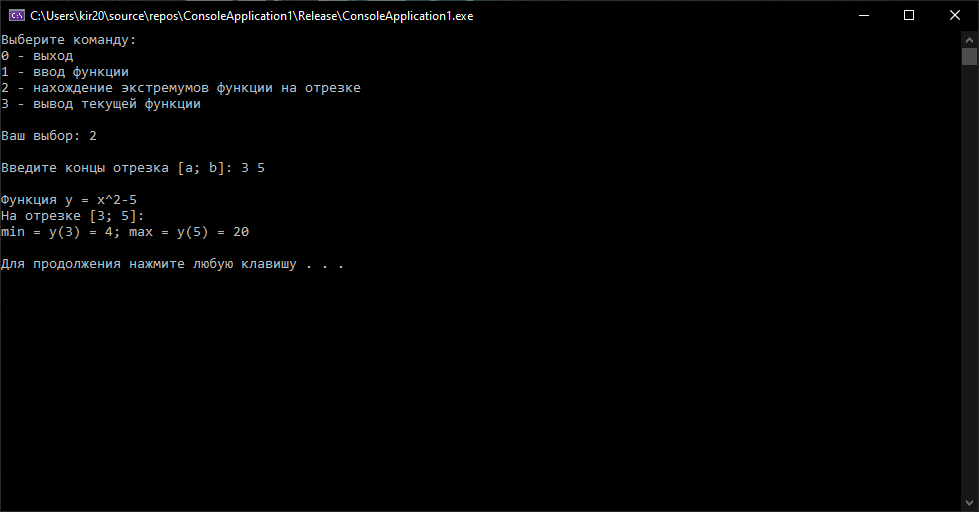


Рисунок 8 «Поиск экстремумов функции на отрезке»

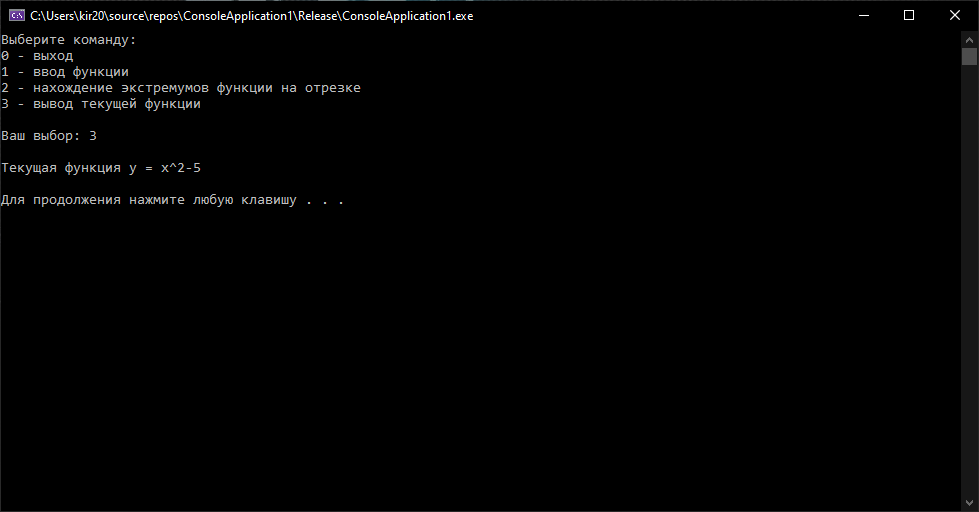


Рисунок 9 «Вывод информации о текущей функции»

Вывод

При разработке программы были изучены средства разработки программ с элементарным интерфейсом консольного режима на языке C++, используемые при структурном подходе, а также схемы алгоритмов, сопровождающие процесс разработки.