МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное

образовательное учреждение высшего образования

«Тверской государственный университет»

Факультет прикладной математики и кибернетики

Направление 02.03.02 – Фундаментальная информатика и

информационные технологии

Профиль «Инженерия программного обеспечения»

Отчет по итогам учебной практики (практики по получению первичных профессиональных умений и навыков)

2017-2018 уч. год 2 семестр

**Автор**: студентка 1 курса 16 гр.

Николаева Мария Сергеевна

**Руководитель практики**:

к.т.н., доцент

Михно Галина Алексеевна

старший преподаватель

Мальцева Людмила Федоровна

**Оценка**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тверь – 2018

**Оглавление**

1. **Функции как формальные параметры**…….………………………..…..3
   1. Реферат на тему…………………………….………………………..….3
   2. Индивидуальное задание………………….………………………..…..3
   3. Алгоритм………………………………….………………………..……4
   4. Исходный код программы……………….………………………..……5
   5. Результаты счета………………………….………………………….…7
2. **Динамические структуры данных**……….………………………………8
   1. Реферат на тему…………………………………………………………8
   2. Индивидуальное задание……………………………………………….9
   3. Алгоритм....………………………………………………………….......9
   4. Исходный код программы……………………………………………...9
   5. Результаты счета……………………………………………………….14

2

1. **Функции как формальные параметры.**
   1. Реферат на тему

Функциональный тип в информатике — тип переменной или параметра, значением которой или которого может быть функция.   
Указатели на функцию используются, когда в программе необходимо производить одинаковые действия или вычисления, незначительно отличающиеся в зависимости от какого-либо условия – например, есть функция вычисления интеграла, которая зависит от f(x), тогда указатель на функцию как раз может передавать эту f(x).

**Объявление указателя на функцию:**  
typedef <type> (\*pointer)(<type>, …, <type>);  
pointer pf;  
pf = &<name\_of\_the\_function>;

Затем pf передается в качестве фактического параметра при вызове функции, как любой другой аргумент.

* 1. Индивидуальное задание (7а)

Задание 1.

7) ,

по формуле прямоугольников:

I  In = h, где fi = f(xi), xi = a + (i-1/2)h, h =  (i = 1, …, n)

и по методу удвоения шагов: | Ι2N - ΙN | < ε

3

1.3. Алгоритм:

● Вычисление интеграла от g(x) по формуле Ньютона-Лейбница   
(double integral\_newton):

1. положить fa =
2. положить fb =
3. вернуть fb - fa

● Вычисление интеграла по формуле прямоугольника

(float integral\_formula\_pryamoug):

1. положить i = 1, I = 0, x = c, h = (d - c) / n
2. пока i ≤ n перейти на шаг 3, иначе перейти на шаг 7
3. положить x = c + (i - 0.5)\*h
4. положить I = I + pf(x), где pf – функция, от которой считается интеграл
5. положить i = i + 1
6. перейти на шаг 2
7. вернуть h\*I

● Вычисление интеграла по методу удвоения шагов

(float integral\_formula\_udv\_shagov):

1. положить n = 10, I = 0, II= 1
2. пока (II - I) ≥ ε перейти на шаг 3, иначе перейти на шаг 7
3. положить I = II
4. положить n = 2 \* n
5. положить II = metod(c, d, n, pf), где metod – ф-ла, по которой необходимо вычислять интеграл
6. перейти на шаг 2
7. вернуть II

4

1.4. Исходный код программы

**main.cpp:**

#include <iostream>

typedef float(\*fx)(float);

typedef float(\*metod)(float, float, int, fx);

float f(float x);

float g(float x);

double integral\_newton(float a, float b);

float first\_integral\_newton(float a, float b);

float integral\_formula\_udv\_shagov(float a, float b, float e, fx func, metod sp);

float integral\_formula\_pryamoug(float a, float b, int n, fx func);

int main()

{

setlocale(LC\_ALL, "Russian");

float a, b, e;

int n;

fx func, fg;

func = &f; fg = &g;

metod sp = &integral\_formula\_pryamoug;

printf("Введите a и b (нижний и верхний соответственно пределы интегрирования\nдля первого интеграла):\n");

printf("\na = "); scanf\_s("%f", &a);

printf("b = "); scanf\_s("%f", &b);

printf("\n\nc = 1, d = 2,5 (нижний и верхний соответственно пределы интегрирования\nдля второго интеграла) заданы в задании\n");

printf("\n\nВведите n - число разбиений отрезка для вычисления по формуле прямоугольника:");

printf("\nn = ");

scanf\_s("%d", &n);

printf("\n\nВведите е - точность вычисления для метода удвоения шагов:");

printf("\ne = ");

scanf\_s("%f", &e);

printf("\n\nF(x)=x по формуле Ньютона-Лейбница: %lf\n", first\_integral\_newton(a, b));

printf("\nF(x)=x по формуле прямоугольника: %f\n", integral\_formula\_pryamoug(a, b, n, fg));

printf("\nF(x)=x по методу удвоения шагов: %f\n\n", integral\_formula\_udv\_shagov(a, b, e, fg, sp));

printf("\nG(x) по формуле Ньютона-Лейбница: %lf\n", integral\_newton(1, 2.5));

printf("\nG(x) по формуле прямоугольника: %f\n", integral\_formula\_pryamoug(1, 2.5, n, func));

printf("\nG(x) по методу удвоения шагов: %f\n\n\n\n", integral\_formula\_udv\_shagov(1, 2.5, e, func, sp));

system("pause");

5

return 0;

}

**functions.cpp:**

#include <iostream>

typedef float(\*fx)(float);

typedef float(\*metod)(float, float, int, fx);

float f(float x)

{

return (1 / (x\*x + 2 \* x + 3));

}

float g(float x)

{

return x;

}

double integral\_newton(float c, float d)

{

double fa, fb;

fa = 1 / sqrt(2) \* atan((c + 1) / sqrt(2));

fb = 1 / sqrt(2) \* atan((d + 1) / sqrt(2));

return fb - fa;

}

float first\_integral\_newton(float a, float b)

{

return b\*b / 2 - a\*a / 2;

}

float integral\_formula\_pryamoug(float c, float d, int n, fx pf)

{

int i = 1;

float h, I = 0, x = c;

h = (d - c) / n;

while (i <= n)

{

x = c + (i - 0.5)\*h;

I = I + pf(x);

i++;

}

return h\*I;

}

float integral\_formula\_udv\_shagov(float c, float d, float e, fx pf, metod metod)

{

int n = 10;

6

float I = 0, II= 1;

while ((II - I) >= e)

{

I = II;

n = 2 \* n;

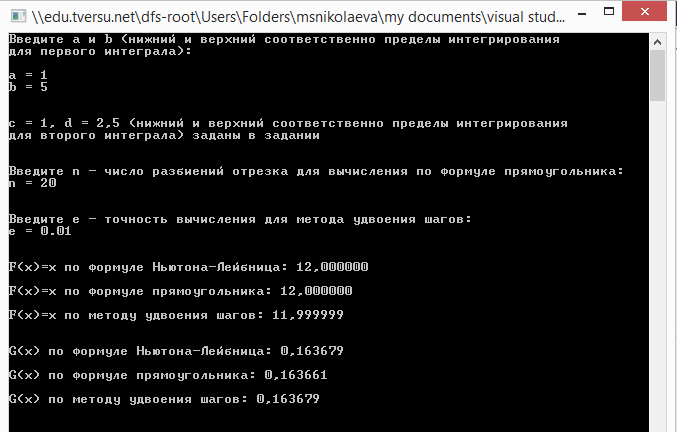
II = metod(c, d, n, pf);

}

return II;

}

1.5.Результаты счета



7

1. **Динамические структуры данных**
   1. Реферат на тему

Динамические структуры используются при работе с данными, размер которых заранее неизвестен или изменяется в процессе выполнения программы.   
 *•* ***Односвязный список:***

Списком называется структура данных, каждый элемент которой посредством указателя связывается со следующим элементом.

Каждый элемент связанного списка, во-первых, хранит какую-либо информацию, во-вторых, указывает на следующий за ним элемент. По такому списку можно двигаться только в одну сторону – от первого элемента к последнему, к концу списка.  
*Объявление односвязного списка:*

struct list

{

<type> info;

list\* next;

};

***• Двусвязный список:***

Отличается от односвязного тем, что его элементы хранят указатель не только на следующий, но и на предыдущий элемент, соответственно, по такому списку можно двигаться в двух направлениях.

*Объявление двусвязного списка:*

struct list

{

list\* prev;

<type> info;

list\* next;

};

Если указатель на следующий элемент в последнем эл-те списка указывает на его первый элемент, то такой список является ***кольцевым***.

***• Стек:***Абстрактная структура данных, работа с которой осуществляется по алгоритму «last in, first out». Элемент может быть взят из стека только из его конца и добавлен так же только в конец. Реализуется с помощью односвязного списка или одномерного массива (при этом экономится память, так как нет нужды хранить указатель на следующий элемент).

***• Очередь:***

Абстрактная структура данных, работа с которой осуществляется по алгоритму «first in, first out». Элемент может быть взят только из начала очереди, а добавлен только в ее конец. Реализуется на массивах (что экономит память, но ограничивает максимальный размер очереди) и на списках (что немного увеличивает используемую память).

8

* 1. Индивидуальное задание

Задание 2.   
Получить сумму двух длинных натуральных чисел. Каждое число представлено стеком (информационная часть элементов стека - две цифры числа). Результат представлен односвязным линейным списком, информационная часть его элементов аналогична информационной части элемента стека.

* 1. Алгоритм

Сложение двух длинных натуральных чисел:  
(st\* addition):

1. положить \*pt1 = s1, \*pt2 = s2, per = 0, res = NULL
2. пока (pt1 ≠ NULL или pt2 ≠ NULL) перейти на п.3, иначе на п.15
3. если (pt1 ≠ NULL и pt2 ≠ NULL) перейти на п.4, иначе на п.7
4. положить per = pt1 info + pt2 info + per
5. положить s1 = pop(s1), pt1 = s1
6. положить s2 = pop(s2), pt2 = s2
7. если pt1 ≠ NULL перейти на п.8, иначе на п.10
8. положить per = pt1 info + per
9. положить s1 = pop(s1), pt1 = s1
10. положить per = pt2 info + per
11. положить s2 = pop(s2), pt2 = s2
12. положить res = add\_b(res, per % 100)
13. положить per = per / 100
14. перейти на п.2
15. если per ≠ 0 перейти на п.16, иначе на п.17
16. положить res = add\_b(res, per)
17. вернуть res
    1. Исходный код программы

#include <iostream>

struct st

{

int info;

struct st\* next;

};

st\* create(int n)

{

st\* el;

9

el = new st;

el->info = n;

el->next = NULL;

return el;

}

bool prov\_void(st\* head)

{

if (head == NULL)

return false;

else return true;

}

void push(st\* a, int el)

{

st \*pn = new st;

pn->info = el;

pn->next = NULL;

if (a != NULL) pn->next = a;

a = pn;

}

st\* add\_b(st\* first, int n)

{

st\* pn = new st;

pn->info = n;

pn->next = NULL;

if (first != NULL) pn->next = first;

return pn;

}

st\* add\_end(st\*first, int n) {

st\*pr, \*pt, \*pn;

pn = new st;

pn->info = n;

pn->next = NULL;

if (first == NULL) return pn;

pt = first; pr = NULL;

while (pt != NULL) {

pr = pt;

pt = pt->next;

}

pr->next = pn;

return first;

}

10

st\* pop(st \*a)

{

st \*pt = a, \*pn = new st;

pn = pt->next;

pt = NULL;

return pn;

}

st\* fill\_stack(st\* t, int n, FILE\* f, bool fl)

{

char c, c1;

st\* head = t; //

if (!fl)

{

c1 = fgetc(f);

head = create(c1 - 48);

for (int i = 1; i < n; i++)

{

c = fgetc(f);

c1 = fgetc(f);

if (c1 == '\n') head = add\_b(head, c - 48);

else head = add\_b(head, ((c - 48) \* 10 + c1 - 48));

}

return head;

}

c = fgetc(f);

c1 = fgetc(f);

if (c1 == '\n') head = create(c - 48);

else head = create((c - 48) \* 10 + c1 - 48);

for (int i = 1; i < n; i++)

{

c = fgetc(f);

c1 = fgetc(f);

if (c1 == '\n') head = add\_b(head, c - 48);

else head = add\_b(head, ((c - 48) \* 10 + c1 - 48));

}

return head;

}

void pechat(st\* head, int r)

{

st\* cur;

cur = head;

int i = 0;

while (cur != NULL)

{

11

if (cur->info < 10) printf("0%d", cur->info);

else printf("%d", cur->info);

cur = cur->next; i = i + 2;

}

printf("\n\n");

}

st\* addition(st\* s1, st\* s2)

{

st\* pt1, \*pt2; int per = 0;

pt1 = s1; pt2 = s2;

st\* res = NULL;

while (pt1 || pt2)

{

if (pt1 && pt2)

{

per = pt1->info + pt2->info + per;

s1 = pop(s1);

pt1 = s1;

s2 = pop(s2);

pt2 = s2;

}

else if (pt1)

{

per = pt1->info + per;

s1 = pop(s1);

pt1 = s1;

}

else

{

per = pt2->info + per;

s2 = pop(s2);

pt1 = s2;

}

res = add\_b(res, per % 100);

per = per / 100;

}

if (per != 0) res = add\_b(res, per);

return res;

}

void pechattt(st\* first)

{

st\* cur;

cur = first;

printf("Список: \n");

12

while (cur != NULL)

{

if (cur->info < 10) printf("0%d ", cur->info);

else printf("%d ", cur->info);

cur = cur->next;

}

printf("\n\n");

}

int main()

{

setlocale(LC\_ALL, "Russian");

st \*a = NULL, \*b = NULL;

FILE \*f;

int n = 0, m = 0, i, r;

char c; bool nech = true, nech1 = true;

st \*res = NULL;

if (fopen\_s(&f, "Text.txt", "r") != NULL)

{

printf("Ошибка при открытии файла! Сожалеем. \n");

system("pause");

exit(1);

}

c = fgetc(f);

while (c != '\n') //узнаем длину первого входного числа

{

c = fgetc(f);

n++;

}

c = fgetc(f);

while (!feof(f)) //узнаем длину второго входного числа

{

c = fgetc(f);

m++;

}

if (n > m) r = n + 1;

else r = m + 1;

if (n % 2 == 1) //флаг нужен для более удобного помещения входного

{ //числа в стек в зависимости от того, четна длина

nech = false; //этого числа или нет

n = n / 2 + 1;

}

else n = n / 2;

if (m % 2 == 1)

{

nech1 = false;

13

m = m / 2 + 1;

}

else m = m / 2;

fseek(f, 0, SEEK\_SET);

a = fill\_stack(a, n, f, nech);

fseek(f, 2, SEEK\_CUR);

b = fill\_stack(b, m, f, nech1);

fclose(f);

pechattt(a);

printf("\n\n");

pechattt(b);

res = addition(a, b);

printf("\n\nРезультат: ");

pechat(res, r);

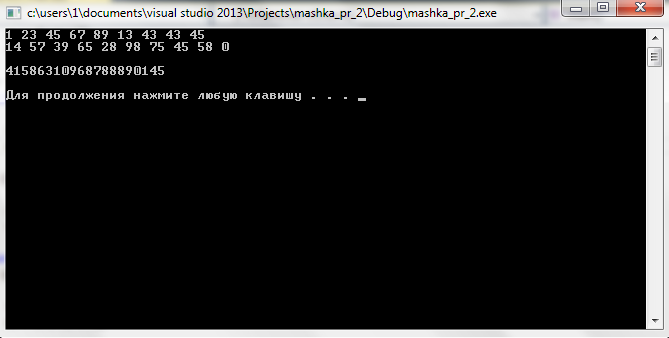
system("pause");

return 0;

}

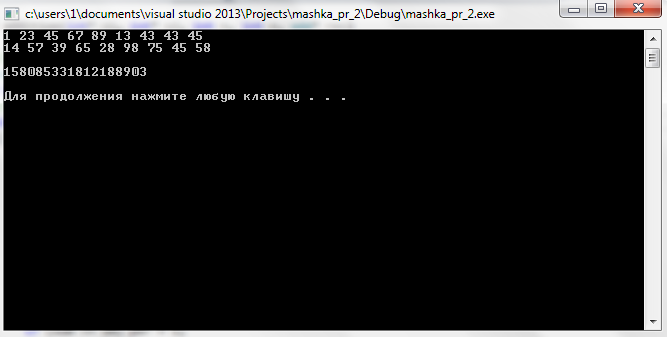
2.5. Результаты счета

Входные числа – 12345678913434345, 14573965289875455800



Входные числа – 12345678913434345, 145739652898754558

14



14