БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
МЕХАНИКО-МАТЕМАТИЧЕСКИЙ ФАКУЛЬТЕТ
Кафедра веб-технологий и компьютерного моделирования

А. С. Кравчук, А. И. Кравчук, Е. В. Кремень

Введение в C++.
Сборник заданий и тематических примеров. Отладка программ

Учебные материалы
для студентов специальности 6-05-0533-07
«Математика и компьютерные науки
(по профилизациям)»

МИНСК
2023

УДК 004.432.045(075.8)
ББК 32.973-018.1я73-1
К78

Рекомендовано советом
механико-математического факультета БГУ
30 мая 2023 г., протокол № 9

Рецензент
профессор кафедры экономической информатики
Белорусского государственного экономического университета
кандидат технических наук, доцент *А. М. Седун*

Кравчук, А. С.

К78 Введение в C++. Сборник заданий и тематических примеров. Отладка программ : учеб. материалы для студентов специальности 6-05-0533-07 «Математика и компьютерные науки (по профилизациям)» / А. С. Кравчук, А. И. Кравчук, Е. В. Кремень. – Минск : БГУ, 2023. – 30 с.

Материалы посвящены основам программирования в C++. В каждой теме приводятся примеры решения типовых задач и варианты индивидуальные заданий. Издание ориентировано в первую очередь на тех, кто не имеет опыта практического программирования на языке C++. Адресуется студентам, а также всем, кто делает первые шаги в программировании.

УДК 004.432.045(075.8)
ББК 32.973-018.1я73-1

Оглавление

Online интегрированные среды разработки. Первая программа. Операции

Интегрированная среда разработки (Integrated development environment, IDE) — комплекс программных средств, используемый программистами для разработки программного обеспечения.

IDE, обычно включает в себя:

- текстовый редактор,
- транслятор (компилятор и/или интерпретатор),
- средства автоматизации сборки,
- отладчик.

Тренд последних лет – создание online или даже облачных интегрированных сред для обучения и даже разработки различным языкам программирования.

Разработчики используют online или облачные IDE для написания, редактирования и компиляции кода непосредственно в браузере, что избавляет их от необходимости устанавливать программное обеспечение на локальные машины. Online и облачные IDE имеют ряд преимуществ перед традиционными (в частности, независимость от используемой разработчиком операционной системы).

Можно просто зайти на соответствующий сайт, добавить или написать свой код и отправить его на компиляцию и выполнить его.

Причины использовать online интегрированные среды:

- требуется быстро проверить идею или алгоритм;
- надо в интерактивную лекцию встроить online пример кода;
- online IDE нужен эпизодически и нет смысла ставить оффлайн аналог на компьютер;
- online IDE часто позволяют сохранить ссылку на выполненный код и передать эту ссылку другому пользователю.

Недостатков в использовании online IDE немного:

- требуется подключение к интернету;
- приходится дольше ждать результат компиляции;
- приходится довольствоваться той конфигураций IDE, которая предусмотрена разработчиками - нельзя добавить свои библиотеки;
- отсутствие или слабая проработанность в подавляющем числе online сред средств отладки.

Однако для учебных целей эти недостатки не существенны. Кроме того, online IDE намного проще в работе чем профессиональные среду, что

позволяет студентам сконцентрироваться на освоении непосредственно языка программирования, а не «выживания» в профессиональной среде.

Простейшая классификация ошибок

В простейшей из известных классификаций ошибок предполагается, что их существуют два вида:

- синтаксические ошибки;
- семантические (логические) ошибки.

В любом языке программирования каждая *инструкция* (*оператор*) строится по определенным правилам. Когда в программе встречается инструкция, которая нарушает эти правила, то говорят о наличии *синтаксической ошибки*. Синтаксическая ошибка легко обнаруживается *компилятором* соответствующего языка программирования и быстро исправляется.

Семантические (логические) ошибки обычно возникает во время выполнения программы, т.е. синтаксис был безупречен. Во многих областях информатики нет абсолютно никакой разницы между *семантической ошибкой* и *логической ошибкой*. Оба означают, что программа скомпилирована, но результат ее выполнения был неверным. Однако в связи с тем, что эти термины плохо определены, то в литературе так же часто они могут означать разные понятия.

Например, если синтаксис программы правильный, но фрагмент кода либо:

- не выполняется,
- инструкции выполняются в *неправильном* порядке,
- логика инструкции неверна,
- код работает с неверными данными,
- непреднамеренно возникает бесконечный цикл,

то разработчик имеет дело с *логической ошибкой*.

К *семантической ошибке* можно отнести случай, когда логика алгоритма и синтаксис программы верны, код выполняется в соответствии с алгоритмом. Однако у некоторых параметров не совпадают размерности.

Таким образом в некоторых пособиях *логическая ошибка* относится к уровню алгоритма программы, а *семантическая* - к ошибке на уровне составления математической модели некоторой предметной области (например, несоответствия размерностей результатов). Но в этом случае, очевидно, что *семантическая ошибка* не имеет ничего общего с программированием.

Поэтому в дальнейшем словосочетания «семантическая ошибка» и «логическая ошибка» будем считать *синонимами*.

Отладка программы

Отладка — это процесс локализации и исправления ошибок в программе.

Отладка бывает двух видов:

- *синтаксическая отладка:* синтаксические ошибки выявляет компилятор, поэтому исправлять их достаточно легко;
- *семантическая (логическая) отладка:* ее время наступает тогда, когда синтаксических ошибок не осталось, но программа выдает неверные результаты.

Запуск интегрированной online среды разработки. Синтаксическая отладка

Замечание.

В связи с большим разнообразием существующих online сред разработки студент в праве выбрать среду, в которой он будет работать самостоятельно.

Не смотря на обилие online интегрированных сред можно предложить две среды на взгляд авторов, наиболее подходящих для учебных целей:

- C++ Shell (http://cpp.sh/);
- OnlineGDB beta (https://www.onlinegdb.com/online_c++_compiler).

Замечание.

Вторая среда OnlineGDB beta из предлагаемых более предпочтительна, т.к. позволяет разрабатывать консольные приложения в том числе для работы с файлами.

Запуск соответствующей интегрированной среды состоит в нажатии мышью на соответствующую интернет-ссылку, приведенную выше, в сочетании с удержанием клавиши `Ctrl`. После чего разработчик попадает на сайт, предполагающий выполнение следующих действий:

- интерактивный ввод текста программ;
- запуск компилятора, а также;
- при отсутствии синтаксических ошибок автоматическое создание и запуск исполняемого кода программы.

Если программа не прошла компиляцию, то это означает, что компилятор нашел синтаксические ошибки в тексте программы и в нижней

части каждой из предлагаемых сред будет открыто окно со списком всех выявленных ошибок.

Для того, чтобы программа была отправлена на выполнение необходимо на первом этапе исправить все ошибки компиляции. Обе среды выдают список ошибок с номерами строк, в которых они допущены.

Однако оформление списка синтаксических ошибок несколько различается в предлагаемых средах:

- в среде C++ Shell, строки с ошибками помечаются красным маркером с андреевским крестом (Рисунок 1).

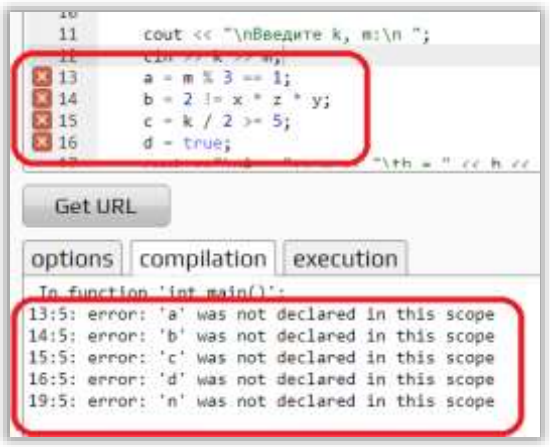


Рисунок 1 – Сообщение об ошибках в C++ Shell

- среда OnlineGDB beta выдает в нижнем окне интерактивный список ошибок (Рисунок 2), кликнув на начало строки красного шрифта, выделенного подчеркиванием, курсор автоматически будет установлен в строке программы, в которой была найдена ошибка.

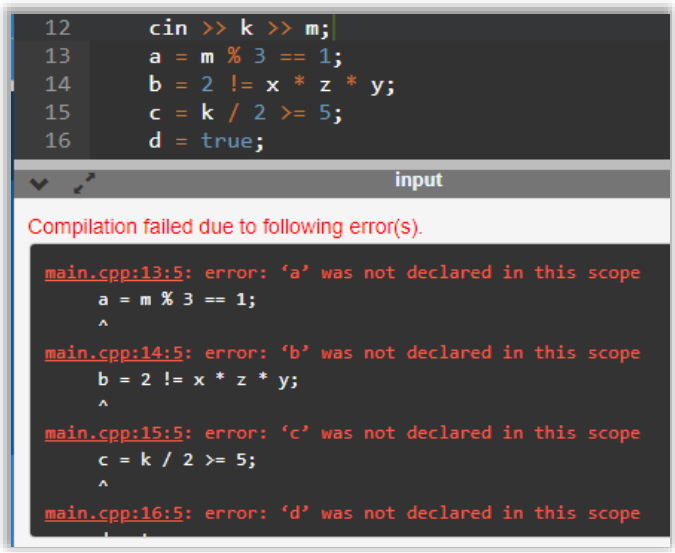


Рисунок 2 - Сообщение об ошибках в OnlineGDB beta

Набор, отладка и запуск первой программы.

Целью выполнения данного задания является обучение студентов правилам организации интерактивного взаимодействия не только с интегрированной средой, но и с простейшей программой, написанной на C++.

Пример.

main.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      char name[20];
6      cout << "Как вас зовут? ";
7      cin.getline (name, 20);
8      cout << "Привет, " << name << "!\n";
9  }
```

Результат работы программы:

```
Как вас зовут? Ренато
Привет, Ренато!
```

Изучение операций в C++

В ходе выполнения задания студент должен ознакомиться с правилами использования операций, их приоритета, а также с преобразованием типов и использования манипуляторов.

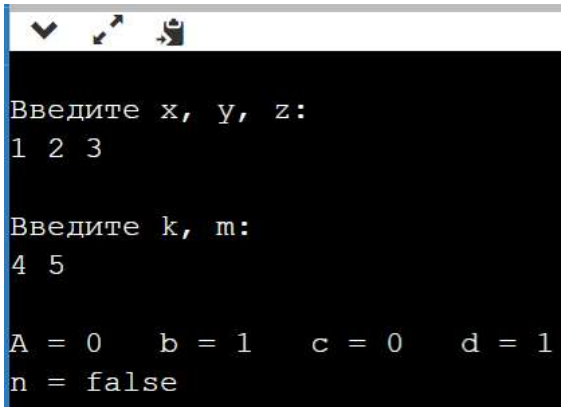
Пример.

main.cpp

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      float x,y,z;
6      cout << "\nВведите x, y, z:\n";
```

```
7       cin >> x >> y >> z;
8       int k,m;
9
10      cout << "\nВведите k, m:\n";
11      cin >> k >> m;
12      bool a, b, c, d, n;
13      a = m % 3 == 1;
14      b = 2 != x * z * y;
15      c = k / 2 >= 5;
16      d = true;
17      cout <<"\nA = "<< a << "\tb = " << b << "\tc = " << c
18           << "\td = " << d;
19      n = !((a || b) && c) == ((! b) && d);
20      // для отображения булевской переменной без
21      // преобразования к int
22      cout << boolalpha
23           << endl << "n = " << n;
24  }
```

Результат работы программы:



Задания:

1. Определить результат определения остатка от целочисленного деления `a % k`, где `a = 100`, а `n` – номер судента в списке подгруппы.
2. Определить результат битового сдвига *вправо* (переменные целочисленные) в выражении `a >> k`, где `a = 256`, а `k = n % 5` (`n` – номер студента в списке подгруппы).
3. Определить результат битового сдвига *влево* (переменные целочисленные) в выражении `a >> k`, где `a = 2`, а `k = n % 5` (`n` – номер студента в списке подгруппы)

4. Вычислить и вывести на экран значения выражений без манипулятора `boolalpha` и с этим манипулятором (разобраться почему результаты вывода на экран в некоторых случаях различаются или совпадают):
   - `((6 % 3 == 1) == (0 == 1));`
   - `((2 != 2*3*4) && (2 != 24));`
   - `((5 / 2 >= 5) || (2 >= 5));`
   - `((8 % 3 == 1) == (2 == 1));`
   - `((2 != 1*1*2) == (!2 != 2));`
   - `5 % 2 ? 2 : 3;`
   - `(7 % 6 ? 0 : 1).`

Программирование линейных и разветвляющихся алгоритмов

Обобщенная формулировка задания. Выдать соответствующее сообщение о принадлежности заданной точки $M(x, y)$ замкнутой области D.

Пример задания. Определить, попадает ли точка внутрь круга, если его центр (`xCircle`, `yCircle`), радиус (`radius`), а также координаты точки (`xPoint`, `yPoint`) вводятся с клавиатуры.

Пример программы

main.cpp

```cpp
#include <iostream>
#include <cmath>
using namespace std;

int main(void) {
    double xCircle, yCircle, radius, xPoint, yPoint;

    cout << "Введите координаты центра круга: ";
    cin >> xCircle >> yCircle;
    cout << endl << "Введите радиус круга: ";
    cin >> radius;
    cout << endl << "введите координаты точки: ";
```

```
13      cin >> xPoint >> yPoint;
14
15      if ( pow(xPoint-xCircle,2)
16           + pow(yPoint-yCircle,2)
17           <= pow(radius,2) ) {
18          cout << endl << "Точка принадлежит кругу";
19      }
20      else {
21          cout << endl << "Точка не попала в круг";
22      }
23      return 0;
24  }
```

Результат работы программы:

```
Введите координаты центра круга: 1 0

Введите радиус круга: 5

введите координаты точки: -6 0

Точка не попала в круг
```

Варианты индивидуальных заданий

Линейные алгоритмы

1. Даны x, y. Получить $\dfrac{|x|-|y|}{1+|xy|}$.
2. Даны два числа. Найти среднее арифметическое кубов этих чисел и среднее геометрическое модулей этих чисел.
3. Вычислить расстояние между двумя точками с данными координатами.
4. Найти площадь треугольника со сторонами a, b, c по формуле Герона: $S=\sqrt{p(p-a)(p-b)(p-c)}$, где $p=\dfrac{(a+b+c)}{2}$.

5. Даны x, y, z. Найти w, если: $w = \sin\left|\left(y - \sqrt{|x|}\right)\left(x - \dfrac{y}{z^2 + \dfrac{x^2}{4}}\right)\right|$
6. Даны x, y, z. Найти w, если: $w = x - \dfrac{x^2}{1 + \sin^2(x + y + z)}$
7. Даны x, y, z. Найти w, если: $w = \cos\left(z^2 + \dfrac{x^2}{4} + y\right)$
8. Даны x, y, z. Найти w, если: $w = y + x - \dfrac{\cos^2 x}{1 + \sqrt{|y + z|}}$

Разветвляющиеся алгоритмы

1. Даны два числа. Вывести первое число, если оно больше второго, и оба числа, если это не так.
2. Найти наименьшее из трех данных чисел.
3. Даны три числа, являющиеся длинами сторон треугольника. Определить тип треугольника (равносторонний, равнобедренный, разносторонний).
4. Даны x, y, z. Найти: $\max\{x + y + z, xyz\} + 3$.
5. Даны x, y, z. Найти: $\min\{x^2 + y^2, y^2 + z^2\} - 4$.
6. Дано x. Вычислить y, если: $y = \begin{cases} x^2, \text{при } -2 \le x \le 2, \\ 4, \text{при } x < -2 \text{ и } x > 2 \end{cases}$
7. Дано x. Вычислить y, если: $y = \begin{cases} 0, \text{при } x \le 0 \\ x, \text{при } 0 < x \le 1, \\ x^4, \text{при } x > 1 \end{cases}$
8. Дано x. Вычислить y, если: $y = \begin{cases} x^2 + 4x + 5, \text{при } x \le 2, \\ \dfrac{1}{x^2 + 4x + 5}, \text{при } x > 2 \end{cases}$
9. Дано x. Вычислить y, если: $y = \begin{cases} 0, \text{при } x \le 0 \\ x^2 - x, \text{при } 0 < x \le 1, \\ x^2 - \sin \pi x^2 - 1, \text{при } x > 1 \end{cases}$

Перечисления enum. Оператор выбора switch()

Обобщенная формулировка задания. С помощью оператора `switch()` и перечисления `enum` вывести на экран соответствующее сообщение.

Пример задания. Дано натуральное (целое, беззнаковое) число `rating`. Вывести строку-описание оценки, соответствующей числу `rating` (`1` - «плохо», `2..3` - «неудовлетворительно», `4..6` - «удовлетворительно», `7..8` - «хорошо», `9..10` - «отлично»). Если `rating` не лежит в диапазоне `1-10`, то вывести строку «`Ошибка ввода!`». Использовать оператор выбора `switch()` и перечисления `enum`.

Пример программы.

main.cpp

```cpp
#include <iostream>
using namespace std;

int main() {

    enum {
        ONE = 1, TWO, THREE, FOUR,
        FIVE, SIX, SEVEN, EIGHT,
        NINE, TEN
    };

    cout << "Введите оценку" << endl;
    unsigned rating;
    cin >> rating;
    switch(rating) {
        case ONE: cout << "плохо" << endl;
                  break;
        case TWO:
```

```
19          case THREE: cout << "неудовлетв."
20                              << endl;
21                      break;
22          case FOUR:
23          case FIVE:
24          case SIX: cout << "удовлетв.";
25                      break;
26          case SEVEN:
27          case EIGHT: cout<< "хорошо";
28                      break;
29          case NINE:
30          case TEN: cout<< "отлично";
31                      break;
32          default: cout<< "Ошибка ввода!"<< endl;
33      }
34      return 0;
35  }
```

Результат выполнения программы:

```
Введите оценку
5
удовлетв.
```

Варианты индивидуальных заданий

Используя оператор выбора `switch()` и перечисления `enum` запрограммировать следующие действия:

1. Дано целое число `k`. Вывести строку-описание оценки, соответствующей числу `k` (`1` - «плохо», `2` - «неудовлетворительно», `3` - «удовлетворительно», `4` - «хорошо», `5` - «отлично»). Если `k` не лежит в диапазоне `1-5`, то вывести строку «ошибка». Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

2. Дан номер месяца - целое число в диапазоне `1-12` (`1` - январь, `2` - февраль и т. д.). Вывести название соответствующего времени года («зима», «весна», «лето», «осень»). Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

3. Ввести номер дня недели и вывести соответствующие ему название дня недели. Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

4. Ввести номер месяца и вывести соответствующее ему название месяца. Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

5. Введите время (только часы). Выведете соответствующее приветствие : «Доброе утро», «добрый день»,» добрый вечер», «доброй ночи». Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

6. Дано число `x` с плавающей точкой. Выведете на экран меню типа (введите номер действия: `1` - возвести число в квадрат; `2` — извлечь корень квадратный; `3` — вычислить синус; `4` - косинус) и выполните эти действия. Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

7. Для натурального числа `k` вывести фразу «`мы нашли k грибов в лесу`», согласовав окончание слова «`гриб`» с числом `k`. Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

8. Дан номер месяца (целое число в диапазоне `1-12`). Вывести на экран количество дней в этом месяце для ***не*** високосного года. Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

9. Вывести на экран значение переменной `k` (от `0` до `10`) римскими цифрами. Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

10. Напишите программу, выводящую сообщение о возможных сообщениях (`1`- фамилия; `2` — имя; `3` — отчество; `4` — год рождения). Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

11. Ввести символ в нижнем регистре от `a` до `f`. Вывести на экран соответствующую букву в заглавном начертании. Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

12. Вывести на экран вместо введенного целого числа `0-9` соответствующее слово. Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

13. Ввести символ в верхнем регистре от `A` до `F`. Вывести на экран соответствующую букву в строчном начертании. Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

14. Дан порядковый номер месяца, вывести на экран количество месяцев, оставшихся до конца года. Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

15. С 1 января этого года по некоторый день прошло `n` дней, определить название текущего месяца. Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

16. Дано число `m` (номер месяца). Определить номер квартала по введенному номеру месяца и номер полугодия. Выдать сообщение о неверном вводе, если введено любое другое значение, кроме оговоренных в задаче.

Оператор цикла `for()`

Обобщенная формулировка задания. Дано натуральное n. Вычислить значение указанной в индивидуальном задании суммы или произведения.

Пример задания. Даны натуральное n. Вычислить значение суммы s по формуле $s = 1 + 2 + \ldots + n$.

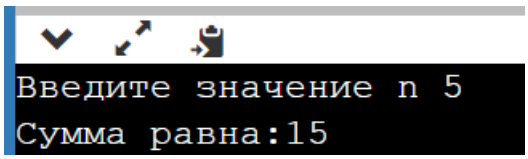
Пример программы.

main.cpp

```
#include <iostream>
using namespace std;

int main(void) {
    int n, sum;

    while (true) {
        cout <<"Введите значение n ";
        cin >> n;
        if (n > 0) break;
        cout << endl << "Введено отрицательное n!!! "
             << endl << "Попробуйте еще раз!!!"
             << endl;
    }

    sum = 0;
```

```
17      for (int i = 1; i <= n; i++) {
18              sum = sum + i;
19      }
20      cout << "Сумма равна:" << sum;
21      return 0;
22  }
```

Результат выполнения программы:



Варианты индивидуальных заданий

Дано натуральное n. Вычислить значение суммы S:

1. $S = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$;
2. $S = 1^2 + 2^2 + \cdots + n^2$;
3. $S = \frac{1}{1^3} + \frac{1}{2^3} + \frac{1}{3^3} + \cdots + \frac{1}{n^3}$
4. $S = 1 + 3 + 5 + \cdots + n$, где n – нечетное число;
5. $S = \frac{1}{1^3} + \frac{1}{3^3} + \frac{1}{5^3} + \cdots + \frac{1}{n^3}$, где n – нечетное число;
6. $S = 2 + 4 + 6 + \cdots + n$, где n – четное число;
7. $S = \frac{1}{2^5} + \frac{1}{4^5} + \frac{1}{6^5} + \cdots + \frac{1}{n^5}$, где n – четное число;
8. $S = 1 \cdot 2 + 2 \cdot 3 + 3 \cdot 4 + \cdots + n \cdot (n+1)$;
9. $S = \frac{1}{1 \cdot 2} + \frac{1}{2 \cdot 3} + \frac{1}{3 \cdot 4} + \cdots + \frac{1}{n \cdot (n+1)}$;
10. $S = 1 \cdot 3 + 3 \cdot 5 + 5 \cdot 7 + \cdots + n \cdot (n+2)$, где n – нечетное число;
11. $S = \frac{1}{1 \cdot 3} + \frac{1}{3 \cdot 5} + \frac{1}{5 \cdot 7} + \cdots + \frac{1}{n \cdot (n+2)}$, где n – нечетное число;
12. $S = 2 \cdot 4 + 4 \cdot 6 + 6 \cdot 8 + \cdots + n \cdot (n+2)$, где n – четное число;
13. $S = \frac{1}{2 \cdot 4} + \frac{1}{4 \cdot 6} + \frac{1}{6 \cdot 8} + \cdots + \frac{1}{n \cdot (n+2)}$, где n – четное число;
14. $S = sin(1) + sin(2) + \cdots + sin(2 \cdot n + 1)$;
15. $S = \frac{1}{sin(1)} + \frac{1}{sin(2)} + \frac{1}{sin(3)} + \cdots + \frac{1}{sin(n)}$;

Оператор цикла `while()`. Табулирование функций

Обобщенная формулировка задания. Вычислить m значений заданной функции $f(x)$ на отрезке $[a,b]$. Результаты оформить в виде таблицы. Значения $a,\ b,\ h$ ввести с клавиатуры.

В таблице:

- первый столбец – узловое значение x_i;
- второй столбец – значения функции $f(x_i)$, вычисленное с использованием библиотечных функций.

Для построения границ строк и столбцов в таблице использовать символы '*', '–', 'I' и т.д.

Пример задания. Построить таблицу значений функции e^x на отрезке [0,1] с шагом h.

Пример программы.

main.cpp

```
#include <iostream>
#include <cmath>
using namespace std;

#define NUM_CHAR 34

int main(void) {
    double a, b, h, x;

    while(true) {
        cout << "Введите a, b, h: ";
        cin >> a >> b >> h;
        if ((a < b) && (h > 0) && (h < b - a)) break;
        cout << endl
             << "Параметры введены неверно!!! "
             << endl
             << "Попробуйте еще раз!!!" << endl;
    }

    cout << "Таблица значенией функции"
         << endl << endl;
```

```
22      //формирование шапки таблицы
23      cout.width(NUM_CHAR);
24      cout.fill('*');
25      cout << ' ' << endl
26           << "*         x       *     значение    *"
27           << endl;
28      cout.width(NUM_CHAR);
29      cout.fill('*');
30      cout << ' ' << endl;
31      //формирование таблицы значений
32      cout.setf(ios::scientific);
33      cout.precision(5);
34      x = a;
35      while(x < b + h / 2) {
36          cout <<"*  " << x << "  *  " << exp(x) << "  *"
37               << endl;
38          x = x + h;
39      }
40      cout.unsetf(ios::scientific);
41
42      //замыкание (подчеркивание) таблицы
43      cout.width(NUM_CHAR);
44      cout.fill('*');
45      cout << ' ' << endl;
46      return 0;
47  }
```

Результат работы программы:

```
Введите a, b, h: 1 2 0.2
Таблица значение? функции

*************************************
*         x       *     значение    *
*************************************
*  1.00000e+00  *  2.71828e+00  *
*  1.20000e+00  *  3.32012e+00  *
*  1.40000e+00  *  4.05520e+00  *
*  1.60000e+00  *  4.95303e+00  *
*  1.80000e+00  *  6.04965e+00  *
*  2.00000e+00  *  7.38906e+00  *
*************************************
```

Варианты индивидуальных заданий

В качестве табулируемой функции $f(x)$ использовать одну из функций:

1. $e^{-x}\sqrt{x}$;
2. $x\sqrt{x}$;
3. $(2x+1)/\sqrt{x}$;
4. $1/\sqrt{x}$;
5. $e^{x^2}/2$;
6. $\sqrt[4]{x}$;
7. $1+e^{x}$;
8. $\sqrt{x^2+1}$;
9. $x\cdot\sin(x)$;
10. $1+\ln^2(x)$;
11. $\cos(5\cdot x)$;
12. $\ln(x)$;
13. $1/(1+e^{-x})$;
14. $\cos(x)e^{-x}$;
15. $x^{3.7}$.

Обработка одномерных массивов

Обобщенная формулировка задания. При выполнении задания предусмотреть возможность введения размерности обрабатываемого массива в диапазоне объявленной максимальной размерности. Исходный массив заполнить случайными целыми числами. Исходный массив и полученный результат вывести на экран.

Примеры задания. Дан одномерный целочисленный массив. Найти и вывести на экран минимальный и максимальный элементы массива.

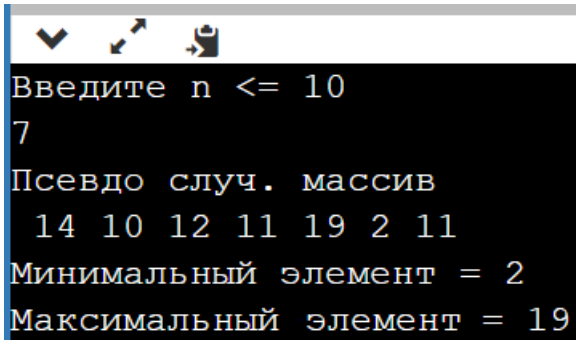
Пример программы.

main.cpp

```
1  #include <iostream>
2  #define MAX_RAND_VALUE 25
3  using namespace std;
```

```
4
5  int main() {
6      const unsigned DIM = 10;
7      int a[DIM];
8      int n;
9      unsigned maxInd, minInd;
10
11     while(true) {
12         cout << "Введите n <= " << DIM << endl;
13         cin >> n;
14         if ((n > 0) && (n <= DIM)) break;
15         cout << "Количество элементов указано не верно."
16              << endl << "Попробуйте еще раз!" << endl;
17     }
18
19     //заполнение массива псевдослуч. числами
20     srand(time(0));
21     for(int i = 0; i < n; i++) {
22         a[i] = rand() % MAX_RAND_VALUE;
23     }
24
25     cout<<"Псевдо случ. массив"<< endl;
26     for(int i = 0; i < n; i++) {
27         cout<<' '<< a[i];
28     }
29
30     maxInd = 0, minInd = 0;
31     for(int i = 0; i < n ; i ++) {
32         if ( a[i] < a[minInd] ) {
33             minInd = i;
34         }
35         else if ( a[i] > a[maxInd] ) {
36             maxInd = i;
37         }
38     }
39     cout << endl
40          << "Минимальный элемент = "<< a[minInd]
41          << endl
42          <<"Максимальный элемент = "<< a[maxInd];
43     return 0;
44 }
```

Результат работы программы:



Варианты индивидуальных заданий

В одномерном массиве из n элементов:

1. найти и вывести на экран порядковые номера и значения первого отрицательного и последнего положительного элементов (если таковые имеются) или выдать соответствующее сообщение об их отсутствии;
2. вычислить сумму всех отрицательных чисел, их количество и сумму всех положительных чисел;
3. в зависимости от того, образуют ли элементы массива целых чисел строго убывающую, невозрастающую, строго возрастающую, неубывающую последовательность, выдать соответствующее сообщение;
4. посчитать количество *положительных* элементов;
5. найти все локальные минимумы и максимумы (элемент называется локальным минимумом (максимумом), если у него нет соседа, меньшего (большего), чем он сам);
6. найти количество элементов, меньших заданного числа и вывести их на экран;
7. посчитать количество *нулевых* элементов;
8. пронормировать его по максимальному элементу (найти максимальный и разделить все элементы массива на найденное значение);
9. посчитать количество элементов, встречающихся только один раз;
10. посчитать количество *отрицательных* элементов.

Семантическая отладка в структурном (процедурном) программировании

В связи с тем, что после выполнения перечисленных выше заданий студенты должны приобрести достаточное представление о программировании простейших алгоритмов на языке C++ и ими уже освоен процесс синтаксической отладки своих программ, то следует перейти к более подробному рассмотрению процесса *семантической* отладки.

Основные принципы применяемые для снижения количества семантических ошибок

Методы (пути) снижения количества ошибок в структурном (императивном, процедурном) программировании:

- применение наиболее простых методов решений поставленных задач;
- использование систем с наименьшим числом параметров;
- использование ранее созданных и проверенных функций;
- привлечение наиболее квалифицированных специалистов;
- проведение тестирования.

Локализация семантических ошибок

Локализация семантических ошибок - это нахождение ориентировочного места семантической ошибки в программе.

В процессе поиска ошибки программист обычно выполняет одни и те же действия:

- «прогоняет» программу и получает результаты;
- сверяет результаты с эталонными и анализирует несоответствие;
- выявляет наличие ошибки;
- выдвигает гипотезу о характере ошибки и ее месте в программе;
- проверяет текст программы;
- исправляет найденную ошибку.

Способы обнаружения ошибки:

- аналитический - имея достаточное представление о структуре программы, просматриваем ее текст и просчитываем фрагменты вручную, без «прогона» на компьютере (таблица ручного счета);

- экспериментальный – «прогоняем» программу на компьютере, используя отладочную печать и/или средства трассировки, и анализируем результаты ее работы.

Оба способа по-своему удобны и обычно используются совместно.

При достаточном опыте большинство ошибок можно обнаружить вообще без запуска программы - просто внимательным просматриванием текста.

Если отладка зашла в тупик и обнаружить ошибку не удается, лучше отложить программу. Когда глаз «замылен», эффективность работы резко падает.

Принципы исправления ошибок больше похожи на законы Мерфи:

- там, где найдена одна ошибка, возможно, есть и другие;
- вероятность, что ошибка найдена правильно, никогда не равна ста процентам;
- исправляя одну ошибку, очень легко внести в программу еще одну или несколько (привнесенные во время исправления алгоритма ошибки являются бичом отладки);
- при отладке и локализации ошибок стоит задача найти саму ошибку, а не ее симптом.

Последнее утверждение необходимо пояснить. Если программа упорно выдает результат `0.1` вместо эталонного нуля, простым округлением вопрос не решить или если результат получается отрицательным вместо эталонного положительного, бесполезно брать его по модулю.

Простейшие средства семантической отладки учебных программ

Чрезвычайно удобные вспомогательные средства - это отладочные механизмы среды разработки, например, *трассировка*. Однако в связи с тем, что учебный курс ориентирован на использование online интегрированных средств разработки, то эти средства в подавляющем большинстве случаев *не*доступны студентам, а когда доступны, то, как правило, очень неудобны.

Таким образом из доступного инструментария отладки в online интегрированных средах остается применение простейшего приема - *отладочной печати* (обычно с помощью `cout`).

Она применяется для:

1. вывода на экран сообщений о ненормальном завершении отдельных блоков, функций и/или всей программы в целом;
2. вывода на экран промежуточных значений параметров в местах, выбранных программистом; обычно, это критичные параметры (индексы, логические переменные) или выражения, от значения которых зависит дальнейший ход выполнения всего алгоритма или, например, составные части сложных формул;

Объект `cerr`

Объект (поток) `cerr` — это объект вывода (как и `cout`), описание которого находится в заголовочном файле `iostream`. С его помощью можно вывести сообщение об ошибке в консоль (как и с помощью `cout`), но только эти сообщения можно еще и перенаправить в отдельный файл с ошибками (`log`-файл). Кроме того, даже если стандартный вывод направлен в файл, то все, что направлено в поток `cerr`, дублируется также и на экране монитора.

Таким образом основное отличие `cerr` от `cout` заключается в том, что `cerr` целенаправленно используется для вывода сообщений об ошибках, тогда как `cout` — для вывода всего остального.

В остальном их синтаксис полностью совпадает.

Использование оператора `return`

Для сигнализации того, что выполнение алгоритма пошло по неверной ветке можно использовать оператор `return`, однако в качестве возвращаемого значения использовать не нуль, а любое целое число. Формат:

```
return неНулевоеЗначение;
```

Замечание.

Поскольку операторов return можно использовать в программе несколько (например, в разветвляющемся алгоритме в конце каждой из веток), то при отладке в качестве возвращаемого целого числа можно возвращать номер ветки, по которой алгоритм закончился.

Использование функции `exit()`

Функция `exit()`, объявленная в заголовочном файле `stdlib.h`, завершает программу на C++. Возвращаемое значение `exit()` передается в качестве аргумента:

```
exit(возвращаемоеЗначение);
```

где `возвращаемоеЗначение` передается операционной системе в качестве кода возврата программы или кода выхода. Принято, чтобы нулевым кодом возврата обозначалось то, что программа завершена успешно.

В другом случае при отладке можно вернуть, например, номер ветки разветвляющегося (или номер цикла циклического) алгоритма, выполнение которой (или которого) привело к завершению программы. Например, два оператора являются эквивалентными:

```
return 3;    или  exit(3);
```

В качестве значения параметра `возвращаемоеЗначение` могут использоваться константы `EXIT_FAILURE` и `EXIT_SUCCESS`, также определенные в заголовочном файле `stdlib.h`. Их можно использовать для обозначения успеха или неудачи выполнения программы.

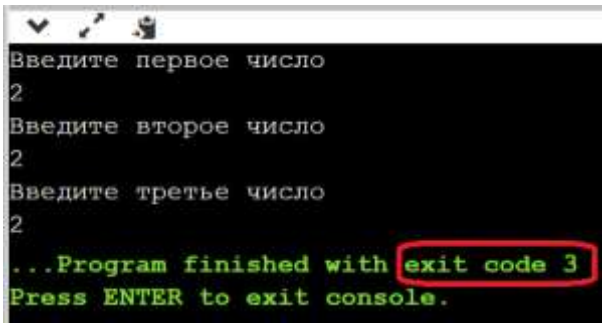
Пример.

main.cpp

```
1  #include <iostream>
2  #include <stdlib.h>
3  using namespace std;
4
5  int main() {
6      double a, b, c;
7      cout<<"Введите первое число"<<endl;
8      cin>> a;
9      cout<<"Введите второе число"<<endl;
10     cin>> b;
11     cout<<"Введите третье число"<<endl;
12     cin>> c;
```

```
13      if ((a > b) && (a > c)) {
14          cout << "max - первое число " << a << endl;
15      }
16      else if ((b > a) && (b > c)) {
17          cout << "max - второе число " << b << endl;
18      }
19      else if ((c > a) && (c > b)) {
20          cout << "max - третье число " << c << endl;
21      }
22      else {
23          //Аварийное завершение выполнения
24          exit(3);
25      }
26      return 0;
27  }
```

Результат работы программы:



Несмотря на то, что функция `exit()` вызывает нормальное завершение программы, но все же, злоупотреблять использованием `exit()` не стоит, поскольку она имеет ряд особенностей, например, когда внутри программы используется функция `exit()`, то деструкторы для локальных объектов не вызываются.

Замечание.

Для завершения программы предпочтительным является использование оператора `return`.

Пример одной из современных классификаций ошибок разработки программ

В интернете можно найти множество классификаций типов ошибок, все они связаны с личным опытом авторов, предложивших ту или иную классификацию. Например, существует следующая классификация:

- *синтаксическая ошибка;*
- *логическая ошибка;*
- *ошибка компиляции:* иногда, синтаксис исходного кода может быть безупречным, но ошибка компиляции все же может произойти. Это может быть связано с проблемами в самом компиляторе либо при использовании online сред разработки в связи с большой нагрузкой на сервер.
- *ошибка среды выполнения:* в случае использования online компиляторов возникает в связи с большой нагрузкой на сервер либо может возникнуть в случае нехватки ресурсов компьютера, на котором установлено программное обеспечение;
- *ошибка ресурса:* переполнение буфера и переполнение стека;
- *ошибка взаимодействия:* может возникнуть в связи с использованием в прикладном программном обеспечении устаревшей версии программного интерфейса приложения (например, использование устаревшего DirectX).

Замечание.

Следует отметить, что существуют и другие классификации ошибок, в основу которых положены иные критерии. В частности, классификация по этапу обработки программы:

- *ошибка компиляции;*
- *ошибка компоновки;*
- *ошибка выполнения;*
- *и т.д.*

Литература

1. Кравчук, А.И. Сборник лабораторных работ и примеров решения задач по алгоритмизации и программированию на языке Си / А.И. Кравчук, А.С. Кравчук - Минск: Технопринт, 2002. - 116 с.
2. Подбельский, В.В. Программирование на языке Си / В.В. Подбельский, С.С. Фомин – М.: Финансы и статистика, 2001. – 600 с.
3. Подбельский, В.В. Язык Си++ / В.В. Подбельский - М.: Финансы и статистика, 2000. – 560 с.
4. Павловская, Т.А. C/C++. Программирование на языке высокого уровня / Т.А. Павловская. - Санкт-Петербург [и др.] : Питер, 2017. - 460 с.

Учебное издание

Кравчук Александр Степанович
Кравчук Анжелика Ивановна
Кремень Елена Васильевна

ВВЕДЕНИЕ В C++. Сборник заданий и тематических примеров. Отладка программ

Учебные материалы
для студентов специальности 6-05-0533-07
«Математика и компьютерные науки
(по профилизациям)»

В авторской редакции

Ответственный за выпуск *Е. В. Кремень*

Подписано в печать 25.08.2023. Формат 60×84/16. Бумага офсетная.
Усл. печ. л. 1,86. Уч.- изд. л. 1,71. Тираж 50 экз. Заказ

Белорусский государственный университет.
Свидетельство о государственной регистрации издателя, изготовителя,
распространителя печатных изданий № 1/270 от 03.04.2014.
Пр. Независимости 4, 220030, Минск.

Отпечатано с оригинал-макета заказчика
на копировально-множительной технике
механико-математического факультета
Белорусского государственного университета.
Пр. Независимости 4, 220030, Минск.