1. Escape-последовательности. К ним относятся:

\n - перенос корректки на слудующую строчку;

\t - табуляция;

\b - удаление аоследнего символа из выводимой строчки;

\\ - выводит один back слэш

\" - выводит двойные ковычки

\' - выводит одинарные ковычки

2. Типы данных

Хранилище данных это оперативная память (ОЗУ(RAM - Random Access Memory))

Числовые

Символьные

Логические

Вещественные числа - это дробные (float(4), double(8))

Целые числа (int(4), shirt(2), long(4))

Символьный - (char(1)) - любые символы на клавиатуре и нетолько

Логический - (bool(1)) - истина лож

3. Переменные

Переменная это облатсть оперативной памяти обладающая собственным именем и предназначенная для

хранения типов данных разных типов - это бирочка или ярлычок по которой мы находим

данные в RAM.

Создание переменной называется специальным термином - ОБЪЯВЛЕНИЕ.

Помещение данных в переменную называется - ПРИСВОЕНИЕ.

Значение переменной может изменятся в пределах одной функции. Название переменной должно нести смысловую

нагрузку.Название не можнт содержать пробел. Можно объявлять сразу несколько переменных через рапятую ->

int a,b,c;

a = 10;

b = 15;

c = 20;

Тип данных char может хранить в себе только один символ;

Тип данных bool может хранить в себе либо "true" or "false", при этом в консоль выводятся 1 или 0 соответственно.

4. Константы

Константа это то что не менятся никогда: колличество дней в неделе, грамм в киллограмме,

Наименования констант проиСходит большими буквами -> COUNT\_DAYS\_IN\_WEEK = 7;

Константы мы менять не можем они статичны!!!

Чтобы не писать каждый раз ESCAPE последовательности можно символ '\n' представить как

константу LIFEHACK

Для переменной типа char ESCAPE последовательность рассматривается как один

символ.

5. Ввод данных

Чтобы что-то работало нужно ввести данные (калькулятор)

Основным инструментом является cin;

Вводить данные можно подряд -> cin >> age >> height >> weight;

6. Операторы. Арифметические операции.

Это конструкция языка позволяющая производить операции с переменными после этих действий появляется результат.

Операторы бывают:

унарные - нужен один оперант; (минус) a = -a;

бинарные - нужны два оператора;(+,-,\*,/,%); преоритет выполнения операций как в матиматике.

умножение деление, а потом сумма и разность.

тернарные

7. Домашняя работа №1

Приведение типов float average = (float)(num1+num2+num3)/3;

Таким образом в ответе может быть число с плавающей точкой.

8. Инкремент и декримент.

Декремент и инкремент имеют две формы префиксная и посфиксная

Если записать сначало переменную, а потом оператор (посфиксно) -> cout << a++ << endl;

то сначало вывидется значение "а" и ++ не сработает, но если (префиксно)-> cout << ++a << endl;

то тогда выведится число на единицу больше!!!

9. Сокращенные арифметические формы +=, -=.

Данное сокращения а += 20 эквивалентны а = а + 20;

Данное сокращение а -= 10 эквивалентно а = а - 10;

10. Логические операции. Операторы сравнения, равенства, объединения, инверсии

Операторы сравоения:

< - меньше

> - больше

<= - меньше или равно

>= - больше или равно

Операторы равенства:

== - равно

!= - не равно

Операторы объединения:

&& - и (коньюнкция)

|| - или (дизъюнкция)

!= - не (инверсия)

Логичкеские операции всегда возвращают булливый тип данных либо true либо false (1 или 0)

В программировании ноль это всегда false, а то что не ноль либо отрицальное число либо положительное будет true

11. Конструкция логического выбора if, ifelse, else.

Женя рассказал про террнарный оператор a ? а : b

Он аналог

if (a){

cout << a <<endl;

}else{

cout << b << endl;

}

string str;

str = (!isRain) ? "Take a umbralla!!! Now is rain!!!" : "No!";

cout << str;

if выполняется если значение в скобках true иначе выполняется значение else.

Проверить числа с плавающей запятой на четность не получиться!!!

12. Switch. Оператор множесвенного выбора.

Если у нас много всяких условий в зависимомти от которых программа должна вести себя

так или иначе, то лучше использовать switch, когда условий не так много то if else.

13. While - это цикл, конструкция

while(утверждение) {

действие для повторения

}

Компилятор выполняет дейсвие и потом возвращается в начало цикла для повторного дейсвия, он это делает до тех пор пока значение утверждения true.

int a = 0;

while(a<10){

cout << " Variable a = " << a << endl;

}

14. do while - это цикл, конструкция

do{ действие

для повторения

} while(утверждение);

15. HomeWork3.1

Задача написать программу которая получает от пользователя три переменные это:

1. Первое это число выводимых элементов;

2. Второе это какой символов выводить;

3. Третье это как выводить (1)горизонтально или (2)вертикально

Далее для реализации использую валидацию правильно ли пользователь ввел число

МОЖНО вводить 1 или 2 все что другое выводит предупреждение

if(count\_symbol != 1 && count\_symbol != 2){

cout << "Don't correct symbol" << endl;

Далее использую do{

if(type\_line ==2){

cout << witch\_use\_symbol << endl;

}

else{

cout << witch\_use\_symbol << " ";

}

a++;

}while (a < count\_symbol);

HomeWork3.2

Програма должна посчитать сумму вссех нечетных элементов из введенного

промежутка

Тогда должно быть три числа: numberStart, namberStop, sum.

do{

if(numberStart%2 != 0){

sum += numberStart}

numberStar++;

} while(numberStart < numberStop);

}

16. Цикл For

for(int i = 0; i < 10; i++){

cout<<"Hello world!!!"<< endl;

}

Так же он будет работать если написать так

for( ; ; ){

cout<<"work cycle"<<endl;

}

Есть еще вариация когда два инкремента i и j

for(int i = 0, j = 10; i < 10; i++, j--){

cout << "I = " << i << endl;

cout << "J = " << j << endl;

}

17. Break - ключевое слово, нужено для выхода из цикла

при исполении какого-то конкретного условия

if(i==7){

break;

}

18. Continue - ключевое слово, нужно для того чтобы пропустить ту или иную

итеррацию, при налчии определенного условия

if(i==7){

continue;

}

19. go to - есть мнение что лучше его вообще не использовать!!!:)

прерывает выполнение программы на любой строчки когда и перейти

в любую другую часть программы.

int main(){

cout << "H" << endl;

goto link;

cout << "L" << endl;

cout << "H" << endl;

link:

cout << "I" << endl;

Cout << "P" << end;

}

Таким образом мы перескочим через

cout << "L" << endl;

cout << "H" << endl;

Сразу в выполнение :

cout << "I" << endl;

НЕ ИСПОЛЬЗУЕТСЯ В 99.999999% СЛУЧАЯХ!!!!

20.Вложенные циклы (Nested Loop)

Они нужны для реализации более мене сложные циклы, как пример сортировка массива.

Моё наблюдение что если объявить два раза переменную i в двух циклах и укать

ее тип данных ( int i в основном и

в ложенном то компилятор будет думать что это две разные переменные, если

во вложеном цикле не указать тип переменной то программа отработает не правильно)

for(int i = 0; i < 5; i++){

cout << i << endl;

for(int i = 0; i < 5; i++){

cout << i << endl;

}

}

28. Sizeof

Помогает узнать размер переменной или массива в байтах, пример кода

int main(){

double arr[]{24,7645,29,2099,90};

cout << sizeof(arr)/sizeof(arr[0]) << endl;

system("pause");

}

29. Random\_generator

Генератция осуществляется благодаря функции rand. При этом числа будут всегда одинковые, для генерации РЕАЛЬНО разных чисел нужно:

1. Применить функцию srand(), передать в качестве аргумента функцию time(NULL)

HomeWork.5.1

Нужно составить массив из уникальных значений

Просто нужно сделать проверку на наличие такого элемента в массиве. Для этого применяется условие:

fore(j = 0; j < i; ){

if(arr[j] == newRandomValue){

bool alreadyThere = true;

breake;

}

}

if(!alreadyThere){

arr[i] = newRandomValue;

i++

}

HomeWork.5.2

Нужно выбрать самое маленькое число в получившемся массиве

Это достигается путем пробега по всему сформированному ранее массиву циклом и сравнение переменной minValue = arr[0] со всеми элементами массива и если элмент меньше то запиши

его значение в переменную minValue:

int minValue = arr[0];

for(i = 1; i < SIZE; i ++){

if(arr[i]<minValue){

minValue = arr[i];

}

}

cout << "Min value in the array: " << minValue << endl;

30. Двумерный массив. Нужно указать сначало значение строки, а потом значение столбца -> arr[1][7] таким образом мы получим координату значения лежащей в этой ячейке с данной координатой

31(35) Передача массива в функцию

32(36). Функция и ее прототипы.

37 Область видимости.

Область видимости это фигурные скобки, если грубо говорить.

Переменные делятся на глобальные и локальные.

Локальные находятся внутри фигурных скобок и за пределами их не видны.

А глобальные переменные находятся не внутри фигурных скобок(функций), а в теле файла программы

Все что внутри фигурных скобок то и изменяется локально, но если изменить глобальную переменную внутри фигурных скобок то ее значение измениться ГЛОБАЛЬНО.

(<https://www.youtube.com/watch?v=a8L1nu4EByI&list=PLQOaTSbfxUtCrKs0nicOg2npJQYSPGO9r&index=47>)

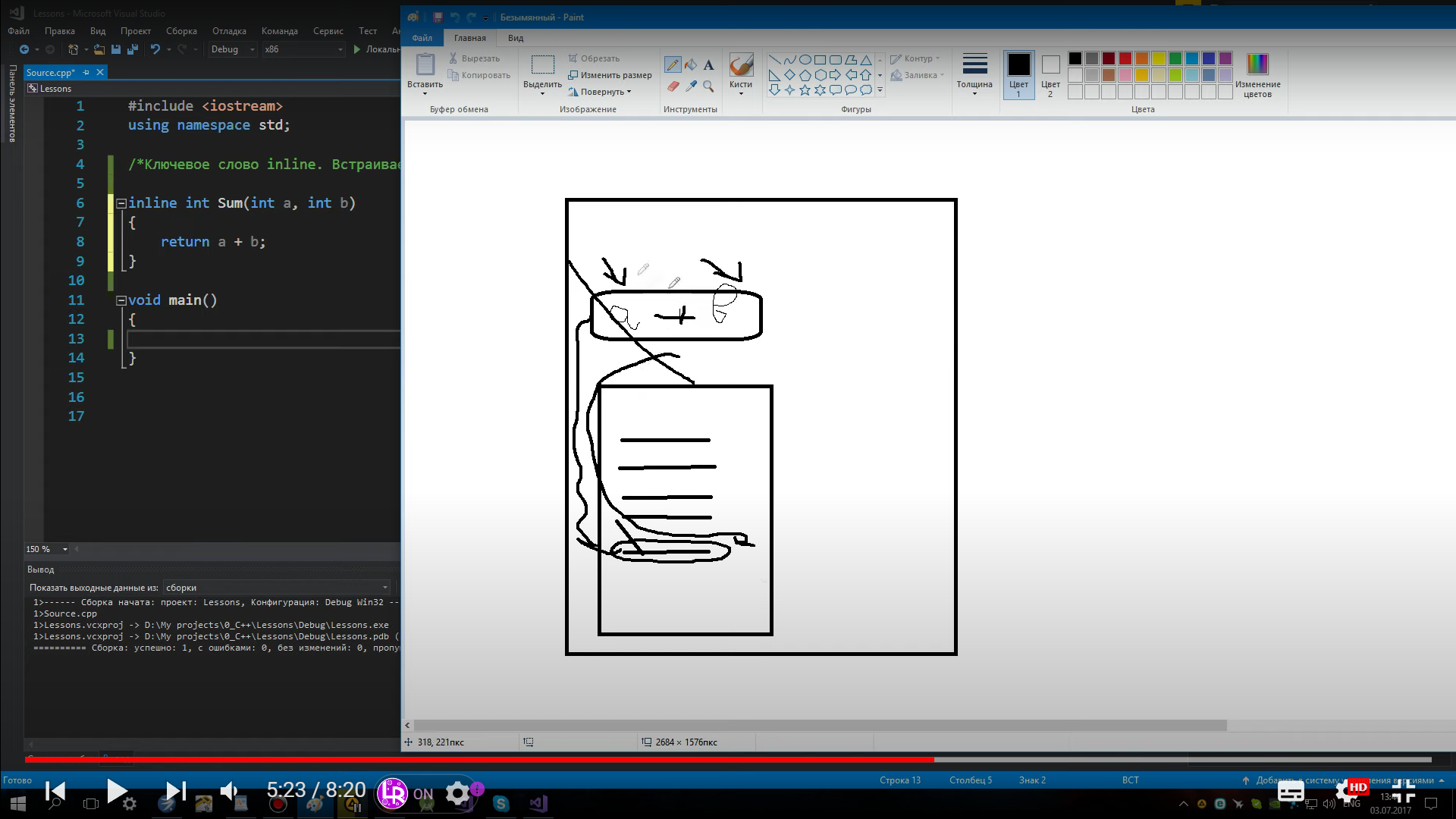
38 Аргументы (Параметры) по умолчанию.

int foo(int f, int a = 10, double d = 10.7)

Параметры без значения по умолчанию должны располагаться в конце объявления параметров.

(<https://www.youtube.com/watch?v=Y0g3cIkCKtw&list=PLQOaTSbfxUtCrKs0nicOg2npJQYSPGO9r&index=48>)

39. Ключевое слово inline. Встраиваемая функция.



Фича inline позволяет ускорить процесс выполнения кода, так как встраивает код реализации функции в следующую строчку после inline.

#40. Перегрузка функций