Задание подготовлено в рамках проекта АНО «Лаборатория модернизации образовательных ресурсов» «Кадровый и учебно-методический ресурс формирования общих компетенций обучающихся по программам СПО», который реализуется с использованием гранта Президента Российской Федерации на развитие гражданского общества, предоставленного Фондом президентских грантов.

**Разработчики**

Лысенко Ирина Владимировна, ГАПОУ «Тольяттинский индустриально-педагогический колледж»

**Назначение задания**

Оценка результата продукта. Оценка программного продукта. Уровень I.

МДК.01.01. Разработка программных модулей

Тема: Структурное программирование. Основы программирования на языке С++

**Комментарии**

Задание предлагается на 2 курсе обучающемуся, который изучил ОПД Основы алгоритмизации и программирования, при изучении новой темы.

Задание подготовлено также в варианте для уровня II

Вы работаете ведущим программистом на языке С++. Ваши коллеги написали программы для дальнейшего использования при обработке больших данных. Вам поручили провести приемку.

Рассмотрите программы, подготовленные коллегами (источник 1). Ознакомьтесь с основами программирования на языке С++ (источник 3) и требованиями к оформлению кода программы (источник 2).

**Оцените соответствие разработанных программ требованиям.**

Заполните бланк.

*Бланк*

**Оценочный лист**

| № | Критерий оценки | Оценка +/- |
| --- | --- | --- |
| Код программы 1 | Код программы 2 | Код программы 3 |
|  | Структура программы соответствует требованиям к структуре C++ |  |  |  |
|  | Подключены необходимые программные модули С++ |  |  |  |
|  | Присутствует главная программа |  |  |  |
|  | Код структурирован |  |  |  |
|  | Присутствуют комментарии |  |  |  |
|  | Реализован диалог с пользователем |  |  |  |
|  | Переменные, в том числе массивы, названы верно |  |  |  |
|  | Переменные названы информативно |  |  |  |
|  | Типы переменных, в том числе массивов, описаны верно |  |  |  |
|  | Константы описаны верно |  |  |  |
|  | Присутствует ветвление |  |  |  |
|  | Цикл со счетчиком оформлен верно |  |  |  |
|  | Присутствует задержка для просмотра результатов выполнения программы пользователем |  |  |  |
|  | Дробные числа появятся на экране в десятичном формате |  |  |  |

***Источник 1***

**Программа 1. Вывести на экран сумму элементов массива n\*n (размеры вводятся с клавиатуры)**

#include <iostream> #include <conio.h>

using namespace std;

float x[100][100], B;

void main() {

cin >> int n ; float S:=0;

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

 cin >> x[i][j];} }

for ( i = 0; i <= n; i++) for ( j = 0; j <= n; j++)

B:=B+ x[i][j];

cout<<B:5:2;

}

**Программа 2. Вывести на экран сумму элементов главной диагонали массива n\*n (размеры вводятся с клавиатуры)**

#include <iostream>

#include <conio.h>

using namespace std;

float x[100][100];

int main()

{

*// Создаем и вводим переменные, которые будут хранить ширину и длину массива*

int n;

cout<< “Введите размер массива ”;

cin >> n ;

*//Вводим элементы массива, i- номер строки, j- номер столбца массива*

for (int i = 0; i <= n; i++)

for (int j = 0; j <= n; j++)

{cout<< “Введите”<< i+1<<”x”<<j+1<<”элемент массива”;

cin >> x[i][j];

}

*//Считаем сумму чисел главной диагонали (у них номер сроки равен номеру столбца)*

float S:=0; *// Обнуляем переменную S для автонакопления суммы в переменную*

for ( i = 0; i <= n; i++)

for ( j = 0; j <= n; j++)

if (j == i) S:=S+ x[i][j];

*//Выводим сумму на экран*

cout<< “Сумма элементов главной диагонали=”<<dec<<S<<endl; ;

getch(); *// Команда задержки экрана*

 return 0;

}

**Программа 3. Вывести на экран сумму четных элементов массива n\*n (размеры вводятся с клавиатуры)**

using namespace std;

float Mas[100][100];

int main()

{

*// Создаем и вводим переменные, которые будут хранить ширину и длину массива*

int number;

cout<< “Введите размер массива- меньше 100”;

cin >> number ;

*//Вводим элементы массива, i- номер строки, j- номер столбца массива*

for (int i = 0; i <= number; i++)

{for (int j = 0; j <= number; j++)

cout<< “Введите”<< i+1<<”х”<<j+1<<”элемент массива”;

cin >> Massiv[i][j];

}

*//Считаем сумму четных чисел (у них остаток от деления на 2 равен нулю)*

int Sum:=0; *// Обнуляем переменную S для автонакопления суммы в переменную*

for ( i = 0; i <= number; i++)

for ( j = 0; j <= number; j++)

{ Ostatok := Massiv [i][j]%2;

if (Ostatok == 0)

Summa:=Summa+ x[i][j];

}

*//Выводим сумму на экран*

cout<< “Сумма четных элементов ”<< Summa;

getch();

return 0;

}

***Источник 2***

**Оформление кода программы**

Обычно, когда мы работаем над конкретным программным продуктом, эстетические качества кода заботят нас далеко не в первую очередь. Нам гораздо важнее наша производительность, качество реализации функционала, стабильность его работы, возможность модификации и расширения и т.д.

Но являются ли эстетические качества кода фактором, положительно влияющим на вышеперечисленные показатели? Ответ: да, и при этом, одним из самых важных!

Это так, потому что красивый код, вне зависимости от субъективной трактовки понятия о красоте, обладает следующими (в той или иной степени сводимыми друг к другу) важнейшими качествами:

* *Читаемость.* Т.е. возможность, глядя на код, быстро понять реализованный алгоритм, и оценить, как будет вести себя программа в том или ином частном случае.
* *Управляемость.* Т.е. возможность в минимальные сроки внести в код требуемые поправки, избежав при этом различных неприятных предсказуемых и непредсказуемых последствий.
* *Минимизация кода*
* *Единообразие* имен переменных, констант, подпрограмм, объектов и т.д., смысловая нагрузка имен (Mashina, Green\_Car) или наличие комментариев для переменных.

Программа, созданная на любом языке программирования, должна соответствовать требованиям не только к работоспособности и функционированию, но и к правильному форматированию (оформлению) кода программы.

Правильно оформленный код является хорошо читаемым, что немаловажно, когда над кодом работает несколько специалистов - программисты, тестировщики, аналитики и программисты во время апгрейдов программ и т.д.

Любую программу намного удобнее читать и понимать, когда в ней:

1. Присутствуют *комментарии* – каждый новый функциональный блок программы, важные программные конструкции, цели создания переменных должны сопровождаться комментариями. Комментарии не являются запускаемыми элементами программы и нужны для описания на обычном языке (русском, английском). Ответственное комментирование намного облегчает понимание кода и повышает возможность его использования.
2. *Код структурирован* - для блоков программы различного уровня вложенности желательно делать сдвиги относительно левого поля. Это улучшает визуальное понимание основных и зависимых конструкций и облегчает чтение кода и логическое понимание выполняемых операций. Также для облегчения чтения программы важно разделять функциональные блоки кода пустыми строками.

Еще одним признаком хорошо структурированного кода является размещение всех операторов блока на одном уровне, в том числе операторных скобок {}.

Пример хорошо структурированного кода с комментариями

// Процедура Sneginka генерирует снежинку на экране в координате 0,100

// Процедура Кaplia генерирует каплю воды на экране в координате 0,0

// Проверка валидности процедуры Sneginka

procedure Sneginka(obj)

*{ if (!is\_valid(obj))*

*return; obj.call1();*

*}*

// Проверка валидности процедуры Sneginka

procedure Kaplia(obj)

*{ if (!is\_valid(obj))*

*return; obj.call2();*

*}*

//key- целая переменная, в которую считывается код нажатой клавиши.

int key;

cin>> key;

//Если нажато 1- то выводим вызываем процедуры снежинки и капли

if ( key==1)

*{ Sneinka(obj);*

*Kaplia(obj);*

*}*

**Пример плохо структурированного кода**

*procedure proc1(obj) { if (!is\_valid(obj)) return; obj.call1(); } procedure proc2(obj) { if (!is\_valid(obj)) return; obj.call2(); } obj = factory.getObject(); if (is\_valid(obj) { proc1(obj); proc2(obj); }*

Также необходимо соблюсти удобство *пользования* программой. Для этого необходимо во время работы программы осуществлять удобный диалог с пользователем, выводить данные в хорошо отформатированном виде. Например. Если вы вводите массив данных, выведите комментарий к пользователю

*cout<< “Введите”<< i<<”x”<<j<<”элемент массива”*

Тогда пользователь увидит фразу в формате

*Введите 2х4 элемент массива*

И поймет, что он вводит в данный момент.

***Источник 3***

**Основы программирования на языке С++**

Cтейтмент (англ. *«statement»*) - это наиболее распространенный тип инструкций в программах. Это и есть та самая инструкция, наименьшая независимая единица в языке С++. Стейтмент в программировании - это то же самое, что и «предложение» в русском языке. Мы пишем предложения, чтобы выразить какую-то идею. В языке C++ мы пишем стейтменты, чтобы выполнить какое-то задание. Все стейтменты в языке C++ заканчиваются точкой с запятой.

Есть много разных видов стейтментов в языке C++. Рассмотрим самые распространенные из них:

*int х*; - это стейтмент объявления (англ. *«statement declaration»*). Он сообщает компилятору, что х является переменной. В программировании каждая переменная занимает определенное число адресуемых ячеек в памяти в зависимости от её типа. Минимальная адресуемая ячейка - байт. Переменная типа int может занимать до 4-х байт, т.е. до 4-х адресуемых ячеек памяти. Все переменные в программе должны быть объявлены, прежде чем использованы. Мы детально поговорим о переменных на следующих уроках.

*х = 5*; - это стейтмент присваивания (англ. *«assignment statement»*). Здесь мы присваиваем значение 5 переменной х.

*cout << x;* - это стейтмент вывода (англ. *«output statement»*). Мы выводим значение переменной х на экран.

*cin>> x; -* это стейтмент ввода

После каждого законченного стейтмента ставится ;

Стейтмент может быть простой командой, а может быть целой веткой условной конструкции, внутри которой размещаются мелкие стейтменты. В связи с этим после команд начала таких конструкций (for, if, while и т.д.) точка с запятой не ставится.

Компилятор также способен обрабатывать выражения. **Выражение** (англ. ***«expression»***) - это математический объект, который создается (составляется) для проведения вычислений и нахождения соответствующего результата. Например, в математике выражение 2 + 3 имеет значение 5/

В языке C++ стейтменты объединяются в блоки - функции.

**Функция** - это последовательность стейтментов. Каждая программа, написанная на языке C++, должна содержать главную **функцию (главную программу) main()**. Именно с первого стейтмента, находящегося в функции main(), и начинается выполнение всей программы.

**Библиотека** - это набор скомпилированного кода (например, функций), который был «упакован» для повторного использования в других программах. С помощью библиотек можно расширить возможности программ. Например, если вы пишете игру, то вам придется подключать библиотеки звука или графики (если вы самостоятельно не хотите их создавать).

Язык C++ не такой уж и большой, как вы могли бы подумать. Тем не менее, он идет в комплекте со **Стандартными библиотеками С++**, которые предоставляют дополнительный функционал. Одной из наиболее часто используемых частей Стандартной библиотеки C++ является **библиотека iostream**, которая позволяет выводить данные на экран и обрабатывать пользовательский ввод

*В начале программы необходимо подключить модули*

*#include <iostream>*

*#include <conio.h>*

Всегда размещайте функцию *main()* в файле .cpp с именем, совпадающим с именем проекта. Например, если вы пишете программу Chess, то поместите вашу функцию main() в файл chess.cpp.

При написании кода нужно указываем компилятору область («тело») функции (например main), подпрограммы, метки, конструкции (например тело цикла, тело ветки условного опеатора и т.д.), с помощью фигурных скобок {…}. Всё, что находится между открывающей фигурной скобкой и закрывающей фигурной скобкой - считается единым «телом».

Если в конструкции цикла или ветке ветвления всего одна команда- скобки не ставятся.

Когда программа завершает свое выполнение, функция main() передает обратно в операционную систему значение, которое указывает на результат выполнения программы: успешно ли прошло выполнение программы или нет. Если оператор return возвращает число 0, то это значит, что всё хорошо! Ненулевые возвращаемые значения указывают на то, что что-то пошло не так и выполнение программы было прервано. Об операторе return мы еще поговорим детально на соответствующем уроке.

Команда *getch()* реализует задержку для просмотра содержимого экрана до нажатия любой клавиши.

**Организация констант и переменных в С++.**

Основополагающим понятием при работе с данными в любом языке программирование является понятие типа данных.

Типы данных для С++ описывают, сколько места в памяти будут занимать ваши данные.

Целое число (int) - одну ячейку в байт, дробное (float)– две, длинное дробное (double) - четыре, символьное - тоже восемь, строковое - столько ячеек, какую длину\количество букв вы указали (а если не указали, то аж 256 ячеек памяти будет зарезервировано под одну строку) и т.д.

**Места в памяти, где хранят данные, называются константами и переменными**

Константы. Иногда в программе бывает так, что во многих командах используется одно и то же число или данное (например, в программе работы с геометрическими фигурами часто нужно значение Пи). Иногда такое значение довольно длинное - и его проще заменить коротким символическим обозначением, иногда вы предполагаете, что его нужно будет поменять во всех командах программы - тогда проще обозначить его по всей программе символическим обозначением, которому присвоить значение один раз в начале программы и т.д. Также необходимо заранее зарезервировать место в памяти для хранения такого значения через тип данных. Такое место с символическим обозначением называется константа.

Переменные. Для того, чтобы программы работали не с одними и теми же данными при каждом запуске программы, а были универсальными для любых вводимых данных (например программа возведения любого числа в любую степень) используются переменные. Переменная - это место в памяти, которому дали имя (имя переменной) и забронировали место (указанием типа переменной). В переенную, в отличие от константы, на протяжении программы можно отправлять разнее данные (менять значение переменной).

Вот пример объявления целочисленной переменной a (которая может содержать только целые числа):

|  |  |
| --- | --- |
|  | *int a;* |

Если переменная дробная (float) , при выводе на экран ее нужно отформатировать из плавающей точки в 10-чное число:

*float S;*

*cout<<S:5:2;*

- задали 5 знаков в целой части, 2 знака в дробной.

Имена в стандартном С++ имеют ограничения - до 8 символов английского языка без знаков препинания и пробелов.

Нужно быть внимательным и следить чтобы имена не менялись в течение программы : вначале программы вы назвали переменную *flag\_massiva*, а ниже уже забыли об этом и называете ее *flag*….

! При описании переменных нужно очень четко следить, какие переменные являются источником данных, какие приемником, и бьются ли у них типы данных.

Например, если вы на зыках Паскаль или бейсик попробуете сложить дробные переменные в целую сумму, то компиляторы укажут вам на ошибку.

В С++ все намного сложнее - итоговое число сложения будет обрезано до целого и помещено в целую переменную суммы без всяких возражений со стороны компилятора. Это может приводить к скрытым ошибкам программ.

Нужно учитывать, что если результат переменной получается из деления (дроби), то она всегда будет дробной.

### Тип *void*

К основным типам языка С++ относится также тип void. Множество значений этого типа пусто. Он применяется для определения функций, которые не возвращают значения, для указания пустого списка аргументов функции, как базовый тип для указателей и в операции приведения типов.

**Создание конструкций в С++**

Конструкциями называются ветвления и циклы в программе.

**Цикл for**

Если мы знаем точное количество действий (итераций) цикла, то можем использовать цикл for. Итерацией цикла называется один проход этого цикла Синтаксис его выглядит примерно так:

*for (счетчик = значение1; счетчик <= значение2; шаг цикла)*

*{*

*тело цикла;*

*}*

Счетчик цикла - это переменная, в которой хранится количество проходов данного цикла. Шаг - значение изменения счетчика. В С++ , в отличие от Паскаля, счетчик и шаг могут быть целым и дробным.

Напишем программу, которая будет считать сумму всех чисел от 1 до 1000.

*#include <iostream>*

*using namespace std;*

*int main()*

*{*

*int i; // счетчик цикла*

*int sum = 0; // сумма чисел от 1 до 1000.*

*setlocale(0, "");*

*for (i = 1; i <= 1000; i++) // задаем начальное значение 1, конечное 1000 и задаем шаг цикла - 1.*

*{*

*sum = sum + i;*

*}*

*cout << "Сумма чисел от 1 до 1000 = " << sum << endl;*

*return 0;*

*}*

**Оператор *if***

Оператор if служит для того, чтобы выполнить какую-либо операцию в том случае, когда условие является верным. *Условная конструкция в С++* всегда записывается в круглых скобках после оператора if.

Внутри фигурных скобок указывается тело условия. Если условие выполнится, то начнется выполнение всех команд, которые находятся между фигурными скобками.

Пример конструкции ветвления

*if (num < 10) { // Если введенное число меньше 10.*

*cout << "Это число меньше 10." << endl;*

*} else { // иначе*

*cout << "Это число больше либо равно 10." << endl;*

*}*

Конструкции ветвления могут быть встроены в циклы, а циклы- в ветвления.

**Массивы.**

**Массив** - это область памяти, где могут последовательно храниться несколько значений.

Массивы - это цепочки (одномерные) или матрицы (двумерные) переменных с одинаковым базовым именем и индексами, обозначающими номер переменной в цепочке или номер строки и столбца.

Возьмем группу студентов из десяти человек. У каждого из них есть фамилия. Создавать отдельную переменную для каждого студента - не рационально. Создадим массив, в котором будут храниться фамилии всех студентов.

Для массива нужно задать тип переменных- для всех будет одинаковым. При описании массива в квадратны скобках указывается максимальное количество элементов, забронировнных в памяти. В самой программе занимать место могут и меньшее количество элементов, а вот большее- нельзя, не «влезающие» элементы в С++ просто будут утеряны.

Нумерация элементов массивов в С++ начинается с 0.

Заполнять массив можно простым перечислением элементов:

*string students[10] = {"Иванов","Петров","Сидоров","Ахмедов", "Ерошкин","Выхин","Андеев","Вин Дизель","Картошкин","Чубайс"};*

Или вносить данные в цикле с клавиатуры или вычислениями.

Для одномерных массивов нужен 1 цикл: для двумерных- 2 цикла, вложенных друг в друга.

*float x[100];*

*for (int i = 0; i <= 99; i++)*

*cin >>x[i];}*

Для двумерных- 2 цикла, вложенных друг в друга ( в примере вводим дробные данные в таблицу 10х10 и считаем сумму чисел в переменную S)

*float x[10][10], S;*

*for (int i = 0; i <= 9; i++)*

*for (int j = 0; j <= 9; j++)*

*cin >> x[i][j];}*

*for (i = 0; i <= 9; i++)*

*for (j = 0; j <= 9; j++)*

*S:=S+ x[i][j];*

*Использованы материалы источников:*

[*https://habr.com/ru/post/266969/*](https://habr.com/ru/post/266969/)

*https://ravesli.com/urok-8-struktura-programm-s/#:~:text=*

*Ответы-,Стейтменты,единица%20в%20языке%20С%2B%2B.*

[*https://code-live.ru/post/cpp-arrays/*](https://code-live.ru/post/cpp-arrays/)*,*

[*https://code-live.ru/post/cpp-if-else/*](https://code-live.ru/post/cpp-if-else/)*,*

[*https://code-live.ru/post/cpp-loops/*](https://code-live.ru/post/cpp-loops/)

Инструмент проверки

| № | Критерий оценки | Оценка +/- |
| --- | --- | --- |
| Код программы 1 | Код программы 2 | Код программы 3 |
|  | Структура программы соответствует требованиям к структуре C++ | + | **+** | + |
|  | Подключены необходимые программные модули С++ | + | **+** | - |
|  | Присутствует главная программа | + | **+** | + |
|  | Код структурирован | - | **+** | + |
|  | Присутствуют комментарии | - | **+** | + |
|  | Реализован диалог с пользователем | - | **+** | + |
|  | Переменные, в том числе массивы, названы верно\* | + | **+** | - |
|  | Переменные названы информативно | - | **-** | + |
|  | Типы переменных, в том числе массивов, описаны верно\* | + | **+** | - |
|  | Константы описаны верно\* | - | **-** | - |
|  | Присутствует ветвление | - | **+** | + |
|  | Цикл со счетчиком оформлен верно | + | **+** | + |
|  | Присутствует задержка для просмотра результатов выполнения программы пользователем | - | **+** | + |
|  | Дробные числа появятся на экране в десятичном формате | + | **-** | - |

*Подсчет баллов*

|  |  |
| --- | --- |
| За каждую полностью верно оцененную программу | 3 балла |
| *За каждую программу, оцененную с одной ошибкой или пропуском*  | *1 балл* |
| ***Максимальный балл*** | ***9 баллов*** |

*\*Пояснения для преподавателя по описанию переменных (в программе 3):*

1) Проблемы с именами переменных. Инициализирован глобальный массив с именем Mas, а используется в программе c именем Massiv Инициализирована переменная Sum, а используется Summa – *переменные названы не верно*.

2) Проблемы с типами данных.

а) Инициализирована целая локальная переменная int Sum, а в нее суммируются дробные числа из массива float Massiv - *переменные описаны не верно*;

б) Переменная остатка Ostatok не инициализирована - не задан тип (причем она должна быть дробная, раз получается делением)- т.е. нет команды float Ostatok - *переменные описаны не верно*.

3) Константы в программе отсутствуют.