Обзор курса и порядок работы студентов в практикуме

Курс «Введение в компьютерные технологии» предусматривает уверенное освоение программирования студентами первого курса. В качестве первого языка выбран один из самых распространенных (и при этом простых) языков программирования – язык программирования Си.

Вспомогательным учебным пособием по синтаксису языка Си может служить:

1. В.А. Антонюк, С.С. Задорожный, А.П. Иванов, А.А. Лукашёв, Н.А. Панов, С.А. Шленов «Язык программирования Си. Учебно-методическое пособие (I семестр).» – М.: Физический факультет МГУ им. М.В.Ломоносова, 2022, 108 с.

Оно имеется в интернете по адресу https://cmp.phys.msu.ru/sites/default/files/MetC_complete.pdf и в компьютерном классе на диске общего доступа.

На семинарах студенты получают теоретические знания по синтаксису и стандартным библиотекам этого языка, а кроме того получают базовые знания по теории вычислений, основным численным методам и часто используемым алгоритмам решения некоторых задач программирования. Обычно семинары проводятся раз в две недели по очередным элементам синтаксиса, после чего выдается очередная задача практикума.

Полученные задания студенты выполняют на практических занятиях в компьютерных классах: используя установленную там среду программирования студенту нужно самостоятельно написать и отладить программу, решающую выданную ему задачу, после чего сдать эту программу преподавателю. Преподаватель во время сдачи задания может задать студенту вопросы по любой части представленной программы, а также попросить при нем выполнить ее небольшую модификацию, чтобы убедиться в том, что студент полностью понимает сдаваемую им программу и способен самостоятельно ее развивать.

В течение семестра студенты должны выполнить 6 заданий, на каждое из которых отводится по 4 часа практикума. В данном пособии не предполагается строгое соответствие между описанными задачами и содержанию заданий. Некоторые главы содержат более одной задачи. Соответствующее задание может тоже содержать несколько задач. В этом случае рекомендуется помещать их решение внутри одной функции *main*() выделив комментариями, где какая задача расположена.

В середине семестра рекомендуется провести коллоквиум в форме тотального опроса всех студентов группы по пройденному материалу и выставить оценку промежуточной аттестации. В конце семестра проводится зачет, который в обязательном порядке включает как ответ на вопросы по теоретическому минимуму языка программирования, так и решение практической задачи, подобной тем задачам, которые студенты решали в течение семестра.

Перед началом работы в практикуме рекомендуется провести устный опрос студентов с целью выяснить какой опыт программирования у них был ранее, на каких языках программирования им приходилось писать программы? И приходилось ли вообще? До сих пор иногда встречаются студенты с очень ограниченными и даже отсутствующими навыками работы даже с самим персональным компьютером, таким студентам нужно уделять больше внимания в практикуме.

Важное замечание: в компьютерных классах практикума может быть установлена более старая версия среды разработки, по сравнению с современными версиями, актуальными в настоящий момент. Однако, для целей учебного процесса первого курса возможностей версии среды разработки, установленной в практикуме, более чем достаточно. При этом мы не приветствуем использование студентами в практикуме первого курса своих ноутбуков с какими-либо другими версиями среды разработки. Причина этого заключается в том, что все студенты должны быть в равном положении (вне зависимости от наличия или отсутствия у них ноутбуков), в том числе они будут должны на зачете продемонстрировать свои знания и навыки именно в той среде, которая установлена в практикуме, а для этого они должны будут получить устойчивые навыки работы в среде практикума в течение семестра. На следующих курсах это ограничение может быть снято.

Часто студенты задают вопрос о разработке ими программ дома. Тут есть некоторая сложность: требования безопасности запрещают студентам подключать какие-либо устройства (включая flash-диски) к компьютерам практикума. Преподаватель по своему усмотрению со своего компьютера может помочь студенту скопировать его исходные тексты из класса на flash-диск студента, но не в обратном направлении! Решением является использование студентом электронной почты: каждый студент получает свой персональный e-mail, доступ к которому возможен через браузер как с компьютера в практикуме, так и с домашнего компьютера студента. Однако, преподаватель должен следить за тем, чтобы студенты понимали сдаваемые ими программы: довольно часто студенты пользуются посторонней помощью при такой удаленной работе, что может привести к большим проблемам на зачете, где такой помощи не будет. И подчеркнем, что работа программы у студента дома вовсе не означает её беспроблемную компиляцию и запуск в классе практикума из-за различий в среде разработке, версии компилятора и поддерживаемых им стандартов языка программирования.

Глава 1. Первая программа.

Особенности языка Си

Программа на языке Си является **структурно ориентированной** и состоит из набора функций и переменных. Функции содержат набор инструкций, описывающих вычисления, которые необходимо выполнить. А переменные хранят значения, которые используются в процессе вычислений.

Структурный подход предполагает алгоритмическое **разбиение задач на подзадачи** и следование следующим правилам:

- Любая программа может быть представлена использованием трёх базовых управляющих конструкций (последовательные операции, ветвление, цикл).
- Повторяющиеся фрагменты оформляются в виде отдельных функций.
- Каждая логически законченная структура оформляется в блок.
- Базовые конструкции могут быть вложены друг в друга.
- Отказ от оператора безусловного перехода *goto*.

Разработчик может определять любые имена для своих функций. Но в программе должна присутствовать функция, с которой начинается исполнение программы. В консольном приложении для нее зарезервировано имя *main*.

Следующие за именем функции круглые скобки предназначены для задания списка параметров (или аргументов), которые передаются в функцию при обращении к ней. Если внутри скобок ничего не указано, то в функцию не передается никаких аргументов. Функция *main* может содержать два параметра, позволяющее передать в функцию текст, который был напечатан в командной строке при запуске программы. Её заголовок тогда выгладит так:

```
int main(int argc, char* argv[])
```

argc определяет количество переданных параметров (если передано только имя программы, то *argc* равно 1), *argv* – массив строковых параметров.

Перед именем функции указывается тип возвращаемого значения. Функция *main()* должна возвращать целочисленное значение (*int*). После завершения программы это значение можно будет увидеть в строке сообщения среды разработки. Для этого в коде перед завершением функции *main()* нужно добавить строку:

```
return 0;
```

Число ноль принято интерпретировать как успешное завершение программы.

Каждый оператор в языке Си заканчивается символом «точка с запятой». В качестве оператора может выступать вызов функции или осуществление некоторых операций. Для написания комментариев используются следующие символы:

`/*` - начало комментария,

`*/` - конец комментария.

При этом все что будет заключено между этими символами будет являться комментариями, то есть не будет восприниматься компилятором как инструкция или набор инструкций. Такое ограничение области комментариев удобно использовать при комментировании блока программы, состоящего из нескольких строк. При необходимости комментария одной строки удобно использовать символы //, при этом комментарием будет являться все, что расположено между символами // и концом строки (т.е. все, что расположено справа от данных символов).

Функции для выполнения своей работы могут вызывать другие функции, которые либо пишутся самим программистом, либо берутся готовыми из имеющихся библиотек. Для подключения объявлений функций из системных библиотек используется команда препроцессора *#include*, например

```
#include <stdio.h>
```

подключает объявления функций стандартной библиотеки ввода-вывода.

Стоит также обратить внимание на возможность использования русских шрифтов при вводе-выводе. Дело в том, что кодировка консольного приложения настроена на использование старой 866 кодовой таблицы символов, которая не совпадает с таблицей windows. Проблему можно решить несколькими способами.

Наиболее простой способ – вызвать функцию установки русской локали:

```
setlocale(LC_ALL, "rus");
```

Однако, у этого способа есть два нюанса. Функция влияет только на вывод, а ввод все равно останется в старой кодировке. И, кроме того, русская локаль в языке Си предусматривает для вещественных чисел указание десятичной запятой, а не десятичной точки.

Другой способ – это использовать консольную команду, устанавливающую для ввода и вывода кодовую страницу windows:

```
system("chcp 1251 > NUL");
```

В этом случае ввод и вывод русских букв будет корректным. Единственное, при первом открытии консоли нужно в ее системном меню выбрать шрифт «*Lucida Console*».

Примеры простых программ на языке Си

Рассмотрим несколько простых примеров программ. Если какие-то из команд будут непонятны, то это сейчас неважно. Они будут рассмотрены в следующих темах.

Пример 1. Сравнение двух чисел.

Программа берет два числа либо из консольной командной строки, если они там есть, либо запрашивает их у пользователя программы и далее определяет, какое из них больше.

```
#include <stdio.h>  // подключение заголовочного файла ввода-вывода
#include <stdlib.h> // подключение заголовочного файла стандартной библиотеки
#include <locale.h> // подключение заголовочного файла локализации

// начало функции main являющейся точкой входа в программу
// параметры функции: argc - количество параметров, переданных в командной строке
// argv - адреса переданных слов
int main(int argc, char* argv[])
{
	int a, b;  // объявление двух переменных типа integer
	setlocale(LC_ALL, "rus"); // для вывода русских строк
	if(argc<3) // проверяем содержимое командной строки
	{
		// в командной строке недостаточно данных
		printf("Введите число 1: "); // запрос на ввод 1 числа
		scanf("%d", &a); //считывание числа в переменную a
		printf("Введите число 2: "); // запрос на ввод 2 числа
		scanf("%d", &b); //считывание числа в переменную b
		getchar();// считывание введенного, но уже ненужного символа enter
	} else
	{
		// числа можно забрать из командной строки
		// читаем в a число из второго параметра командной строки
		a = atoi(argv[1]);
		// читаем в b число из третьего параметра командной строки
		b = atoi(argv[2]);
	}
	if (a > b) // условный оператор, проверяющий верно ли, что a>b
	{
		printf("Первое число (%d) больше второго (%d)\n",a,b);
	}else
	{
		if (a == b) // условный оператор,проверяющий верно ли,что a равно b
		{
			printf("Числа равны(%d)=(%d)\n",a,b);
		}
		else
		{
			printf("Второе число (%d) больше первого (%d)\n",b,a);
		}
	}
	getchar();// ожидаем ввода символа, чтобы терминал не закрылся
	return 0;
} // конец функции main
```

Пример 2. Использование оператора множественного выбора.

Программа выполняет расчет периметра квадрата, площади квадрата или объема куба используя оператор *switch*.

```
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char* argv[])
{
	char b; // объявление символьной переменной
	double a,res; // объявление действительных переменных двойной точности
```

```
	system("chcp 1251 > NUL"); // настройка на ввод и вывод русского текста
	if(argc<2) // проверка наличия числа в командной строке
	{
		// в командной строке нет данных
		printf("Введите длину стороны квадрата: ");
		scanf(" %lf", &a); // ввод длины стороны с клавиатуры
		getchar();// считывание ненужного символа enter
	} else
		a = atof(argv[1]); // ввод числа из командной строки

	printf("Введите параметр расчета: S-площадь, V - объём, P - периметр:");
	scanf(" %c", &b); // вид расчета всегда вводим с клавиатуры
	getchar();// считывание введенного символа enter
	switch(b)
	{
	case 'S':
		res=a*a; // вычисление площади
		printf("Площадь - %lf\n", res);
		break;
	case 'V':
		res=a*a*a; // вычисление объема
		printf("Объем - %lf\n", res);
		break;
	case 'P':
		res=4.*a; // вычисление периметра
		printf("Периметр  - %lf\n", res);
		break;
	default:
		printf("Ошибка\n");
		break;
	}
	getchar();// ожидаем ввода символа, чтобы терминал не закрылся
	return 0;
}
```

Пример 3. Вычисление логических побитовых операций

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>

int main()
{
	unsigned int a, b, c, d, e, f, g, h;
	//объявление беззнаковых переменных типа integer
	setlocale(LC_ALL, "rus"); // переходим в консоли на русский язык
	a = 15;
	b = 136;
	c = ~a;   // not a
	d = a<<1; // сдвиг a на 1 разряд влево
	e = a>>1; // сдвиг a на 1 разряд вправо
	f = a&b;  // a and b
	g = a^b;  // a xor b
	h = a | b;// a or b
	printf("Операция ~: %d\n", c);
	printf("Операция <<: %d\n", d);
	printf("Операция >>: %d\n", e);
	printf("Операция &: %d\n", f);
	printf("Операция ^: %d\n", g);
	printf("Операция |: %d\n", h);
	getchar();
	return 0;
}
```

Пример 4. Расчет суммы чисел от 1 до 100 с использованием цикла.

Программа использует цикл *do{..}while (условие окончания цикла).* Цикл повторяется до тех пор, пока

условие не станет ложным.

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
int main()
{
setlocale(LC_ALL, "rus"); // переходим в консоли на русский язык
	int i,sum; //объявление переменных типа int
	i=1;		// начальное значение i равно 1
	sum=0;		// начальное значение sum рвно 0
	do //повторять
	{
		sum += i; // увеличить sum на i
		i++;		// увеличить  i на 1
	}while (i<101); // пока значение i не станет равным 101
	printf("Сумма чисел от 1 до 100 = %d", sum);//вывод суммы
	getchar();// ожидаем ввода символа, чтобы терминал не закрылся
	return 0;
}
```

Пример 5. Поиск максимума во введенной последовательности целых чисел.

```
#include <stdio.h>
#include <stdlib.h>
#include <locale.h>
int main()
{
	int a,amax;
	setlocale(LC_ALL, "rus"); // переходим в консоли на русский язык
	printf("Введите первое число последовательности:\n");
	scanf(" %d",&a);
// начальное значение максимума берем равным первому введенному числу:
	amax = a;
	printf("Вводите числа последовательности(0 - окончание ввода):\n");
//повторять, пока нет ошибок ввода и введенное число не равно нулю:
	while (scanf(" %d",&a)==1 && a!=0)
	{
		if (amax < a)// сравниваем очередной введенный элемент и макс.
			amax = a; // берем больший из них
	}
	getchar(); // убираем символ ENTER
	printf("Максимум = %d\n", amax);
	getchar();// ожидаем ввода символа, чтобы терминал не закрылся
	return 0;
}
```

Варианты задач для решения.

Для выполнения задач необходимо создать консольный проект и добавить в него один новый *task1.cpp* файл. Необходимые для ввода данные нужно читать из командной строки, если их окажется недостаточно, то запрашивать у пользователя (аналогично тому, как сделано в первом примере). При отладке программы из Visual Studio данные командной строки нужно записать в командные аргументы проекта (команда меню Проект–>свойства–>свойства конфигурации –> отладка). Подробно эти действия описаны в главе 2 пособия «Язык программирования Си».

1. Вариант Дано целое положительное число. Написать программу, проверяющую истинность высказывания: «Данное число является четным двузначным».
2. Вариант Даны три целых числа: A, B, C. Написать программу, проверяющую истинность высказывания: «Каждое из чисел A, B, C положительное».
3. Вариант Написать программу, спрашивающую у пользователя несколько чисел (больше трёх) и выводящую их среднее значение и дисперсию (среднее квадрата отклонения от среднего значения).

4. Вариант
Написать программу, проверяющую, что, либо A четно, либо B четно (но не оба одновременно).

5. Вариант
Написать программу, спрашивающую у пользователя три числа и выводящую их в порядке возрастания.

6. Вариант
Даны три целых числа: A, B, C. Написать программу, проверяющую истинность высказывания: «Справедливо двойное неравенство A < B < C».

7. Вариант
Даны два целых числа: A и B. Написать программу, проверяющую истинность высказывания: «Числа A и B имеют одинаковую четность».

8. Вариант
Написать программу, проверяющую, что среди чисел A, B, C и D есть как минимум 2 нечетных.

9. Вариант
Используя оператор switch, написать программу вывода прописью значения введенной переменной, если ее значение в диапазоне от 1 до 5. В остальных случаях значение вывести цифрами.

10. Вариант
Даны два целых числа A и B (A < B). Написать программу, выводящую в порядке убывания все целые числа, расположенные между A и B (не включая числа A и B).

11. Вариант
Даны два целых числа A и B (A < B). Написать программу, находящую произведение всех целых чисел от A до B включительно.

12. Вариант
Дано целое число N (> 1). Написать программу, находящую наименьшее целое число K, при котором выполняется неравенство 3K > N.

13. Вариант
Даны координаты трех точек на плоскости. Написать программу, вычисляющую значение линейной интерполяции или экстраполяции для любого введённого x по двум уравнениям прямых, проведенных через соседние точки.

14. Вариант
Написать программу, вычисляющую площадь треугольника по трем сторонам. Если треугольник с такими сторонами не существует, то вывести сообщение об этом.

15. Вариант
Написать программу, проверяющую является ли прямоугольным треугольник с заданными сторонами. Если треугольник с такими сторонами не существует, то вывести сообщение об этом.

16. Вариант
Написать программу, вычисляющую площадь треугольника по заданным координатам его вершин.

17. Вариант
Написать программу, поверяющую является ли равносторонним треугольник с заданными координатами его вершин.

18. Вариант
Написать программу, подсчитывающую сумму всех нечетных чисел от 1 до указанного N.

19. Вариант
Написать программу подсчитывающую количество десятичных цифр введенного натурального числа.

20. Вариант
Известно, что сумма N первых нечетных чисел равна квадрату числа N, например, $1 + 3 + 5 = 3^2$, $1 + 3 + 5 + 7 = 4^2$ и т.д. Написать программу, выводящую таблицу всех натуральных чисел от 1 до K и их квадратов, вычисленных с помощью указанного соотношения.

21. Вариант
Написать программу возведения числа в степени 2,3,4 или 5 используя оператор switch.

22. Вариант

Написать программу, вычисляющую значение факториала $N! = 1 \cdot 2 \cdot 3 \cdot \ldots \cdot N$.

23. Вариант

Написать программу расчета суммы чисел в диапазоне от n до N. Числа n и N должны вводиться с клавиатуры.

24. Вариант

Написать программу вывода всех чисел в диапазоне от 0 до указанного N кратных 3 или 5.

25. Вариант

Написать программу расчета суммы введенной последовательности чисел. Первым вводимым параметром задаётся количество чисел. Если числа заданы в командной строке, то их количество равно *argc* – 1.

26. Вариант

Написать программу проверки того, что элементы введенной последовательности чисел не убывают. Первым вводимым параметром задаётся количество чисел. Если числа заданы в командной строке, то их количество равно *argc* – 1.

27. Вариант

Написать программу, которая выводит попарные суммы вводимой последовательности чисел. Первым вводимым параметром задаётся количество чисел. Если числа заданы в командной строке, то их количество равно *argc* – 1.

28. Вариант

Написать программу, которая вычисляет разность между средним и минимальным значениями во вводимой последовательности. Первым вводимым параметром задаётся количество чисел. Если числа заданы в командной строке, то их количество равно *argc*–1.

29. Вариант

Написать программу, которая во вводимой последовательности определяет количество чисел равных заданному значению (задать его в виде константы). Первым вводимым параметром задаётся количество чисел. Если числа заданы в командной строке, то их количество равно *argc*–1.

30. Вариант

Написать программу суммирования всех десятичных чисел в диапазоне от 1 до указанного N, у которых последняя цифра больше 5.