ПЗ-01: **Программирование разветвляющихся алгоритмов**

1. Операторы условных и безусловных переходов.
2. Инструкция **if-else**.
3. Инструкция **switch**.

В программах с разветвленной структурой используются условные операторы, которые предназначены для выбора к исполнению одного из нескольких возможных действий (операторов) в зависимости от некоторого условия (при этом одно из действий может отсутствовать - пустой оператор).

|  |  |
| --- | --- |
| **Условный оператор** используется, если необходимо выполнять действия в зависимости от выполнения или невыполнения какого-то условия.  Для организации условных и безусловных переходов в программе на языке Си используются операторы: **if - else, switch** и **goto**.  Первый из них записывается следующим образом:  **if (проверка\_условия) оператор\_1; else оператор\_2;**  Если условие в скобках принимает истинное значение, выполняется **оператор\_1**, если ложное – **оператор\_2**.  Если вместо одного необходимо выполнить несколько операторов, то они заключаются в фигурные скобки.  В операторе **if** слово **else** может отсутствовать.  В операторе **if-else** непосредственно после ключевых слов **if** и **else** должны следовать другие операторы.  Если хотя бы один из них является оператором **if**, его называют ***вложенным***.  Согласно принятому в языке **Си** соглашению слово **else** всегда относится к ближайшему предшествующему ему **if**.  **Задача 1**. *Даны два числа A и B. Вывести большее из них*. | http://inf-w.ru/wp-content/uploads/2021/03/%D0%B2%D0%B5%D1%82%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D0%B51-1.png |
| **Условная инструкция**  Синтаксис условной инструкции выглядит следующим образом:  ***if (выражение\_1)***  ***{***  ***// Инструкции\_1, если выражение в условии***  ***// принимает значение "истинно" или не равно 0***  ***}***  ***[else***  ***{***  ***// Инструкции\_2, если выражение в условии***  ***// принимает значение "ложь" или равно 0***  ***}]*** | **Вложенные инструкции if**  В тех случаях, когда в программе двух ветвей условной инструкции недостаточно, прибегают к вложенным ветвлениям.  Вложенное ветвление реализуется условной инструкцией, которая в одной (или в обеих) из ветвей содержит другую условную инструкцию. |
| Поскольку ветвление предполагает максимум два пути решения, то часто приходится вставлять одно ветвление внутрь другого, то есть применять вложенные ветвления. Частным случаем вложенного ветвления является переключатель (switch-case).  Оператор **switch** позволяет выбрать одну из нескольких альтернатив. Он записывается в следующем формальном виде:   |  | | --- | | ***switch (выражение)***  ***{***  ***case константа\_1: операторы\_1; break;***  ***case константа\_2: операторы\_2; break;***  ***default: операторы\_default;***  ***}*** |   **Задача 2**. *Дано числовое представление дня недели: 0 – воскресенье, 1 – понедельник, 2 – вторник и т. д. Вывести, соответствующее ему, текстовое значение.*  Переключатель (**switch**) работает следующим образом. *Выражению-селектор* в заголовке **switch** передается *значение целочисленного объекта*. Это значение по очередности сопоставляется со всеми значениями блоков **case** переключателя. Если значение какого-либо блока **case** окажется равным значению селектора, то выполняется инструкция этого блока и выход из **switch**.  В противном случае (если будут пропущены все блоки **case**), выполняется блок по умолчанию (**default**), а если и он отсутствует, то switch передает управление в программу.  В различных языках программирования ветвления реализованы похожим образом, в виде инструкции управления  **if-else** или **swith-case** (для переключателя). | http://inf-w.ru/wp-content/uploads/2021/03/%D0%BF%D0%B5%D1%80%D0%B5%D0%BA%D0%BB%D1%8E%D1%87%D0%B0%D1%82%D0%B5%D0%BB%D1%8C.png |

**Пример**:

|  |  |
| --- | --- |
| Команда компиляции для программы на Cи выглядит следующим образом:  **gcc -Wall -o prog1 prog1.c** | *#include <stdio.h>*  *#include <stdlib.h>*  *#define show(){printf("x=%lf",x);}*  *void main(void){*  *double a,b,x;*  *printf("a=");*  *scanf("%lf",&a);*  *printf("b=");*  *scanf("%lf",&b);*  *if (a>b) {x=a/b+31;show();exit(0);}*  *if (a==b) {x=-25;show();exit(0);}*  *if (a<b)*  *{x=(a\*5-1)/a;*  *show();*  *exit(0);*  *}*  *}* |

где:

**gcc** — компилятор для языка программирования Cи;

**-Wall** — ключ вывода всех предупреждений компилятора;

**-o prog1** — с помощью ключа -o указывается имя выходного файла;

**prog1.c** — имя нашего исходного файла, который компилируем.

Выполнив команду **ls** (или **ls** **–l**), увидим, что появилась наша скомпилированная программа **prog1**, отмеченная цветом, обозначающим исполняемые программы.

В завершение запустим **prog1**, вводом имени программы с префиксом **./**: **./prog1**

Префикс **./** сообщает терминалу о необходимости выполнить программу с заданным именем в текущем каталоге (Точка – это условное название текущего каталога)/

|  |  |
| --- | --- |
| **Булева алгебра**  Иногда необходимо проверить сразу несколько условий.  Например, чтобы выйти на улицу, нужно убедиться, что нет дождя или у нас есть зонт. Чтобы завести машину, нам нужны ключ и бензин в баке.  Такие условия называются сложными, а для их проверки применяется булева алгебра. В Cи для этого есть следующие операторы:  **&&** — логическое «И» или логическое умножение (конъюнкция).  Оператор И возвращает истину, если верны оба утверждения.  **||** — логическое «ИЛИ» или логическое сложение (дизъюнкция).  Оператор ИЛИ возвращает истину, если верно хотя бы одно утверждение.  **!** — логическое отрицание. Возвращает истину, если утверждение ложно. |  |

**Задачи:** Решить задачи на Си (Linux, CLI), нарисовать блок-схемы решения задач (в тетради).

**Задача 1**. Найти максимальное число из трех:

Вводятся три целых числа. Определить какое из них наибольшее.

a, b, c – переменные, которым присваиваются введенные числа.

**Задача 2**. Вычислить значение функции y=f(x)

Дана следующая функция y=f(x): y = 2x — 10, если x > 0; y = 0, если x = 0; y = 2 \* |x| — 1, если x < 0

Требуется найти значение функции

**Задача 3**. Определить является ли число N кратным числу K.

**Задача 4**.

|  |  |
| --- | --- |
| Написать программу вычисления стоимости покупки с учетом скидки.  Скидка в 3% предоставляется, если сумма покупки больше 500 руб, в 5% — если сумма больше 1000 руб. | Ниже приведен рекомендуемый вид экрана программы:  **Вычисление стоимости покупки с учетом скидки**  **Введите сумму покупки и нажмите <Enter>-> 640**  **Вам предоставляется скидка 3%**  **Сумма с учетом скидки: 620.80 руб.** |

**Задача 4**.

|  |  |
| --- | --- |
| Написать программу проверки знания даты основания Санкт-Петербурга.  В случае неправильного ответа пользователя, программа должна выводить правильную дату. | Ниже приведен рекомендуемый вид экрана программы  **В каком году был основан Санкт-Петербург?**  **Введите число и нажмите <Enter>-> 1705**  **Вы ошиблись, Санкт-Петербург был основан в 1703 году** |

**Задача 5**.

Написать программу, которая проверяет, является ли введенное пользователем целое число четным.

**Задача 6**.

Написать программу, которая вычисляет оптимальный вес человека, сравнивает его с реальным и выдает рекомендацию о необходимости поправиться или похудеть.

Оптимальный вес вычисляется по формуле: **Рост (см) – 100**.

**Задачи на switch**

– Дан порядковый номер дня недели, вывести на экран его название

– Дан порядковый номер карты от 6 до 14, определить ее достоинство. Достоинства определяются по следующему правилу: туз -14, король - 13, дама -12, валет -11, десятка - 10, ... , шестерка – 6

– Дан признак транспортного средства: а - автомобиль, в - велосипед, м - мотоцикл, с - самолет, п - поезд. Вывести на экран монитора максимальную скорость транспортного средства в зависимости от введенного признака.