Практическое занятие 8. Условные операторы и опер цикла

Задание 1. Условные операторы.

Цель - изучить конструкцию "**if-else**".

1. В домашнем каталоге создаем и переходим в папку "**TASK_08_01**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_08_01 <Enter>
$cd TASK_08_01 <Enter>
$
```

2. Создаем файл "**cndtn.c**" и добавляем в него следующие строки:

```
#include <stdio.h>

#define PASSWD 1234 // Константа с паролем

int main(int argc, char *argv[]) {
  int passwd = 4321;

  // Сравниваем значение переменное passwd с ожидаемым значением,
  // заданным макроопределением PASSWD
  if(passwd == PASSWD){
        printf("Welcome\n");

        return 0;
  }

  prinf("Access denied!\n");

  return 1;
}
```

3. Самостоятельно собрать/запустить программу. Объяснить наблюдаемый вывод на экран.

4. Самостоятельно в проверке условия заменить логический оператор равенства ("==") на оператор присвоения ("="). Собрать/запустить программу. Объяснить наблюдаемый вывод на экран.

ЗАМЕЧАНИЕ При замене логического оператора компилятор даже не выдал предупреждения. Для исключения подобных ситуаций в будущем, можно в условии поменять местами константу и переменную. Вместо "**passwd == PASSWD**"писать "**PASSWD == passwd**". Использование оператора присвоения в такой конструкции приведет к ошибке компиляции и мы сразу увидим проблемное место в коде.

5. Самостоятельно модифицировать/собрать программу так, чтобы на пароль считывался с клавиатуры.

ЗАМЕЧАНИЕ Здесь и далее все изменения в рамках задания стараться использованием условной компиляции.

Задание 2. Оператор выбора.

Цель - изучить конструкцию "**switch-case-default**".

1. В домашнем каталоге создаем и переходим в папку "**TASK_08_02**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_08_02 <Enter>
$cd TASK_08_02 <Enter>
$
```

2. Создаем файл "**switch.c**" и добавляем в него следующие строки:

```
#include <stdio.h>

int main(int argc, char *argv[]) {
    int day = 5;

    switch (day) {
    case 0:
        printf("Sunday\n");
        break;
    case 1:
        printf("Monday\n");
        break;
    case 2:
        printf("Tuesday\n");
        break;
    case 3:
        printf("Wednesday\n");
        break;
    case 4:
        printf("Thursday\n");
        break;
    case 5:
        printf("Friday\n");
        break;
    default:
        printf("Saturday\n");
    }

    return 0;
}
```

3. Самостоятельно собрать/запустить программу. Объяснить наблюдаемый вывод на экран.

4. Самостоятельно в программе убрать все операторы "**break**". Собрать/запустить программу. Объяснить наблюдаемый вывод на экран.
5. Самостоятельно модифицировать/собрать программу так, чтобы день недели считывался с клавиатуры.
6*. Самостоятельно модифицировать/собрать программу так, чтобы день недели передавался как аргумент программе.

ЗАМЕЧАНИЕ Для преобразования строкового параметра в целое можно воспользоваться функцией "**sscanf**()" или "**atoi**()". Более подробно см. встроенную справку.

Задание 3. Циклы.

Цель - изучить конструкции языка Си для организации циклов.

1. В домашнем каталоге создаем папку "**TASK_08_03**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_08_03 <Enter>
$cd TASK_08_03 <Enter>
$
```

2. Создаем файл "**while.c**" с кодом программы, которая с шагом h = 0.1 в диапазоне от $-x_0 = 1$ до $x_1 = 1$ будет выводить на экран значение функции $f(x) = x^2$:

```
#include <stdio.h>

#define F(X) (X*X) // Опишем функцию при помощи макроса

int main(int argc, char **argv) {
  double x0 = -1.0; // Левая граница
  double x1 = 1.0;  // Правая граница
  double h = 0.1;   // Шаг по x
  double x = x0;    // Текущее значение x
  double f = F(x0); // Текущее значение функции

  // Пока x меньше x1 "вертимся" в цикле
  while(x < x1){
        f = F(x);
        printf("f(%f) = %f\n", x, f);
        x = x + h; // увеличиваем x на h

  }

  return 0;
}
```

3. Самостоятельно собрать/запустить программу. Объяснить наблюдаемый вывод на экран.

4. Самостоятельно, не меняя функциональности, модифицировать программу так, чтобы цикл был организован при помощи конструкции "**do-while**".

5. Самостоятельно, не меняя функциональности, модифицировать программу так, чтобы цикл был организован при помощи "**for**".

6*. Самостоятельно написать программу для численного вычисления интеграла функции (например, $f(x) = x^3$). Пределы интегрирования задавать через аргументами программы.