Практическое занятие 7. Базовые типы данных, переменные, указатели, форматированный ввод

Задание 1. Базовые типы данных.

Цель - проанализировать размер базовых типов данных и их вариаций.

ВАЖНО! Стандарт С89 определяет пять фундаментальных типов данных: "**char**" (символьные данные), "**int**" (целые), "**float**" (с плавающей точкой), "**double**" (двойной точности), "**void**" (без значения).

1. В домашнем каталоге создаем и переходим в папку "**TASK_07_01**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_07_01 <Enter>
$cd TASK_07_01 <Enter>
$
```

2. Создаем файл "**types081.c**" и добавляем в него следующие строки:

```
int main(int argc, char **argv) {
   int i;
   int sz0, sz1;

   // вычисляем размер в байтах переменной i
   sz0 = sizeof(i);

   // вычисляем размер в байтах типа int
   sz1 = sizeof(int);

   return 0;
}
```

3. Самостоятельно модифицировать/собрать программу так, чтобы на экран выводились значения размера переменной "**i**" и типа "**int**". Запустить программу и проанализировать вывод на экран.
4. Самостоятельно вывести на экран размер остальных основных типов Си и их производных ("**signed char**", "**unsigned char**", "**short int**", "**long int**", "**long long int**" и пр.).

Задание 2. Область видимости локальных переменных.

Цель - изучить область видимости переменных и проанализировать возможные ошибки при "маскировании" локальных переменных.

1. В домашнем каталоге создаем и переходим в папку "**TASK_07_01**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_07_02 <Enter>
$cd TASK_07_02 <Enter>
$
```

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

```
#include <stdio.h>

int main(int argc, char **argv) {
  int var = 5;
  printf("1. var = %d\n", var);

  {
        int var = 500;
        printf("1. var = %d\n", var);

  }

  printf("1. var = %d\n", var);

  return 0;
}
```

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

ВАЖНО! При написании программ подобные ситуации нужно избегать!

Задание 3. Глобальные переменные.

Цель - познакомиться с глобальными переменными.

1. В домашнем каталоге создаем папку "**TASK_07_03**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_07_03 <Enter>
$cd TASK_07_03 <Enter>
$
```

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

```
#include <stdio.h>

int glb_var = 5; // Заводим глобальную переменную

// добавляем функцию, увеличивающую глобальную переменную на единицу
void my_inc(){
  glb_var++;
}
```

```
int main(int argc, char **argv) {

  printf("1. glb_var = %d\n", glb_var);

  my_inc(); // вызываем функцию увеличивающую glb_var на единицу.

  printf("2. glb_var = %d\n", glb_var);

  return 0;
}
```

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

4. Самостоятельно перенести "**my_inc()**" в отдельный файл (не забываем создавать заголовочные файлы). Для доступа к глобальной переменной **glb_var** в новом файле необходимо добавить строку:

```
extern int glb_var;
...
```

5. Самостоятельно собрать/запустить программу. Убедиться, что все работает.

6. Самостоятельно переписать программу без использования глобальных переменных (модифицировать функцию "**my_inc()**" так, чтобы ей можно было передать аргумент, а сама функция возвращала бы увеличенное на единицу значение аргумента).

ВАЖНО! При написании программ использование глобальных переменных должно быть сведено к минимуму!

Задание 4. Указатели.

Цель - познакомиться с указателями.

1. В домашнем каталоге создаем папку "**TASK_07_04**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_07_04 <Enter>
$cd TASK_07_04 <Enter>
$
```

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

```
#include <stdio.h>

int main(int argc, char **argv) {
  int a = 5; // Инициализируем переменную типа int
```

```
  int *ptr = 0x0; // Инициализируем переменную указатель
                  // на переменную типа int

  printf("1. ptr = 0x%X\n",ptr); // Выводим на экран текущее значение
                                 // указателя (он равен 0x0)

  ptr = &a; // Присваиваем указателю адрес, по которому лежит
            // переменная a
  printf("2. ptr = 0x%X\n", ptr); // Выводим на экран текущее значение
                                  // указателя

  printf("3. *ptr = %d\n", ptr); // Выводим на экран значение, которое
                                 //лежит по адресу ptr

  a++; // Изменим значение переменной a
  printf("4. *ptr = %d\n", ptr); // Выводим на экран значение, которое
                                 //лежит по адресу ptr

  return 0;
}
```

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

Задание 5. Библиотечные функции. Форматированный ввод.

Цель - познакомиться с форматированным вводом информации при помощи функции "**scanf()**".

Описание функции "**scanf**()"

Функция "**scanf**()" предназначена для считывания данные из потока "**stdin**". "**scanf**()" может считывать данные всех базовых типов и автоматически конвертировать их в нужный внутренний формат. Если бы "**printf**()" выполняла ввод, а не вывод, ее можно было бы назвать аналогом "**scanf**()". Формат передаваемых "**scanf**()" параметров аналогичен параметрам функции "**printf**()", за исключением того, что в качестве аргументов передаются не переменные, а указатели на них (то есть адреса в памяти, куда необходимо положить считанные значения).

Функция "**scanf**()" возвращает число, равное количеству полей, для которым успешно присвоены значения.

1. В домашнем каталоге создаем и переходим в папку "**TASK_07_05**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_07_05 <Enter>
$cd TASK_07_05
$
```

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

```
#include <stdio.h> // В файле stdio.h задекларирована функция scanf()
```

```
int main(int argc, char **argv) {
  int i; // Заводим локальную переменную типа int
  int ret;

  ret = scanf("%d", &i); // Считываем из стандартного потока переменную
                         // типа int. Если все прошло успешно, то
                         // scanf() вернет 1 (одно присвоение), если
                         // нет, то ноль

  if(ret == 1){
        printf("i = %d\n", i); // Выводим на экран считанное значение
  }else{
        printf("Ошибка!\n");
        return 1;
  }

  return 0;
}
```

3. Самостоятельно собрать/запустить программу. Проанализировать выполнение программы.
4. Самостоятельно модифицировать программу таким образом, чтобы считывать с клавиатуры значение типа double.