Именованные константы. Директива define

Давайте снова обратимся к программе подсчёта чисел, сгенерированных функцией rand:

Листинг 1.

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

int main(void) {
  srand(time(NULL));
  int count[3] = {0};
  int rand_number;

  for (int i = 0; i < 100000; i = i + 1){
    rand_number = rand()%3;
    count[rand_number] = count[rand_number] + 1;
  }

  for(int i = 0; i < 3; i = i + 1){
    printf("%d - %d\n", i, count[i]);
  }

  return 0;
}
```

Допустим, мы хотим расширить диапазон чисел, которые генерируют программа. Вместо чисел от **0** до **2** мы хотим генерировать числа от **0** до **10**.

Для того чтобы внести в программу подобное изменение, нам необходимо заменить **3** на **11** в трёх местах в программе: в объявлении массива **count**, в вызове функции **rand** и в цикле, который выводит массив **count** на экран. Не очень-то удобно, особенно если таких мест будет очень много. Легко запутаться и что-нибудь пропустить.

Чтобы как-то автоматизировать подобный процесс, можно использовать директиву препроцессора **#define**. С её помощью можно создавать так называемые именованные константы.

Синтаксис директивы define

Листинг 2.

```
#define имя значение
```

Важное замечание!
В конце не нужно ставить **;**. Между именем и значением не нужно ставить **=**.

Надеюсь, вы помните, что программа компилируется в машинный код с помощью компилятора? На самом деле этот процесс довольно сложный, да и компилятор состоит из нескольких частей. Одна из частей компилятора – препроцессор. Он обрабатывает текст программы и подготавливает его к компиляции. Одна из функций препроцессора – подключение файлов, указанных в директивах **include**. Другая функция – это обработка инструкций **define**.

С директивой **define** препроцессор работает следующим образом. В тексте программы находятся все места, где встречается имя, указанное в директиве **define**. После этого вместо этого имени подставляется значение, указанное в директиве **define**.

Подстановка значения не производится, если имя встретилось внутри имени переменной, в строке или в комментарии.

Давайте посмотрим на нашем примере.

Листинг 3.

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

#define EXPERIMENTS 1000000
#define NUMBERS 3

int main(void) {

  srand(time(NULL));
  int count[NUMBERS] = {0};
  int rand_number;

  for(int i = 0; i < EXPERIMENTS; i = i + 1){
    rand_number = rand()%NUMBERS;
    count[rand_number] = count[rand_number] + 1;
  }

  for(int i = 0; i < NUMBERS; i = i + 1){
    printf("%d - %d\n", i, count[i]);
  }

  return 0;
}
```

Как видите, я добавил в программу две именованные константы.

Программисты на Си обычно называют именованные константы используя буквы в верхнем регистре, чтобы в коде они сразу бросались в глаза и их было легко отличить от переменных.

После того как мы запустим процесс компиляции, над программой начнёт работать препроцессор. Он подключит три заголовочных файла. После этого вместо имён **EXPERIMENTS** и **NUMBERS** произойдёт подстановка значений **1000000** и **3**, соответственно.

После окончания работы препроцессора этот кусочек программы будет выглядеть вот так:

Листинг 4.

```
int main(void) {

  srand(time(NULL));
  int count[3] = {0};
  int rand_number;

  for(int i = 0; i < 1000000; i = i + 1){
    rand_number = rand()%3;
    count[rand_number] = count[rand_number] + 1;
  }

  for(int i = 0; i < 3; i = i + 1){
    printf("%d - %d\n", i, count[i]);
  }

  return 0;
}
```

В примере выше мы используем именованные константы в качестве, так сказать, настроек (параметров) программы. Меняя значения этих параметров мы изменяем саму программу. Например, меняя значение константы **NUMBER** мы изменяем количество случайных опытов. Это стандартный приём в программировании на Си. Использование данного приёма позволяет писать более понятный код, который, кроме того, будет легко изменить под условия конкретной задачи.

Дополнительные материалы

1. В программировании есть так называемые антипаттерны программирования. Это такие приёмы написания программ, которых стоит избегать. Один из таких приёмов – использование магических чисел. Подробнее об этом здесь. Использование именованных констант – один из способов борьбы с магическими числами.

Интернет версия: http://youngcoder.ru/lessons/8/direktiva_define_i_konstanty.php