НРТК

C / C++

Тема 04. Функции. Начальные сведения

Что такое функция?

- Программа — форма представления алгоритма
- Алгоритм должен допускать декомпозицию
- Функция — часть программы (часть алгоритма программы)
- Каждая функция должна:
  - иметь имя для ее вызова
  - получать параметры (исходные данные)
  - содержать описание алгоритма
  - возвращать результат
- Для применения функций надо уметь:
  - описывать функции
  - вызывать функции

Описание функции

- Синтаксис:

```
<тип> <имя_функции>(<параметры>)
{
  <тело функции>
  return <возвращаемое значение>;
}
```

- где:
  - `<параметры>` — список исходных данных (формальных параметров) в виде:
    `<тип> имя_переменной`
  - `<тело функции>` — описание алгоритма функции
  - `<возвращаемое значение>` — результат, возвращаемый из функции, типа `<тип>`

Примеры

- Умножение двух чисел:

```
double mlt(double p1, double p2)
{
  double mlt = p1 * p2;
  return mlt;
}
```

- Минимум из двух чисел:

```
double min(double a, double b)
{
  if ( a < b ) return a;
  else         return b;
}
```

- Операторов `return` может быть несколько, но не стоит этим слишком увлекаться!

Пример: вычисление НОД

```
int gcd(int a, int b)
{
    int t;

    while ( a != 0 )
    {
        t = a;
        a = b % a;
        b = t;
    }

    return b;
}
```

Вызов функции

- Синтаксис:

  <имя_функции> (<передаваемые параметры>)

- где:
  - **<передаваемые параметры>** — список передаваемых исходных данных (фактических параметров); элементом списка может быть:
    - константа
    - переменная
    - выражение

Вызов функции

- В C++ количество фактических параметров вызова функции и их типы должны совпадать с количеством и типами формальных параметров описания
- Пример вызова функции:

```
int main()
{
  double a = 3.4, b = 8.6, c, d;
  c = mlt(a, b);
  d = mlt(3.7, c) + mlt(a, c - 2.9);
  return 0;
}
```

Итак, программа на C/C++

- Набор функций, которые могут вызывать друг друга
- Среди которых есть одна функция `main`, которую не может вызывать никакая другая функция, с нее начинается исполнение программы
- Функции не могут быть вложены друг в друга

Еще пример

```
double sum(double p1, double p2)
{
  return p1 + p2;
}
double mlt(double p1, double p2)
{
  return p1 * p2;
}
double sum_mlt(double p1 , double p2 , double p3)
{
  return mlt(p1, sum_mlt(p2, p3));
}
int main()
{
  double a = 3.4, b = 8.6, c, d;
  c = mlt(a, b);
  d = sum(3.7, c) * sum_mlt(a, b, -2.9);
  return 0;
}
```

Прототип функции

- В C++ функция должна быть описана до ее вызова
- Прототип функции — предварительное объявление функции
- Синтаксис: объявление тела функции, но тело функции заменено на ; (точку с запятой):

```
<тип> <имя_функции>(<параметры>);
```

- Имена параметров могут быть опущены, но не стоит этим увлекаться: удачные имена функции и параметров — половина документации к функции

Пример использования прототипа

```
double Sum(double, double);
int main()
{
  double a = 3.4, b = 8.6, c;
  c = Sum(a, b);
  return 0;
}

double Sum(double p1, double p2)
{
  return p1 + p2;
}
```

Особенности функций в C

- Не проверяются типы и количество формальных параметров
- Прототип необязателен, если его нет — подразумевается, что функция возвращает `int`
- Хорошие программы на C не используют эти возможности (следовательно, их программы «совместимы» с C++)

Пример

```
double sum();
int main()
{
  double a = 3.4, b = 8.6, c;
  int i;
  c = sum(a, b);     // Нет проверки параметров
  i = gcd(10, 15);   // Вызов без прототипа
  return 0;
}

double sum(double p1, double p2)
{
  return p1 + p2;
}
```

Задачи

- Вычисление периметра и площади треугольника
- Задачи для самостоятельной работы