**Структурний підхід до розробки алгоритмів. Функції.**

При розв’язанні складних задач застосовується структурний підхід до розробки алгоритмів за методикою «зверху-вниз». Цей підхід полягає у розбитті складної задачі на більш прості підзадачі та передбачає розв’язання цих підзадач з використанням функцій.

При розробці консольних програм у С# метод Main є статичним, тому будь-які функції, що можуть бути використані у консольних программах, також повинні бути статичними.

|  |  |
| --- | --- |
| **Загальна форма** | **Приклад**. Знайти максимальне з трьох цілих чисел |
| **static <тип функції> <ім’я функції>(<список форм. парам.>)**  {  //тіло методу  } | **static int Max(int c1,int c2,int c3)**  **{**  **int m = c1;**  **if (c2 > m)**  **m = c2;**  **if (c3 > m)**  **m = c3;**  **return m;**  **}** |

Проілюструємо застосування структурного підходу на прикладі. Обчислити значення виразу 

|  |  |
| --- | --- |
| Без функцій | З функціями |
| using System;  using System.Text;  namespace ConsoleApplication3  {  class Program  {  static void Main(string[] args)  {  int a,b,c;  Console.Write("a=");  a = int.Parse(Console.ReadLine());  Console.Write("b=");  b = int.Parse(Console.ReadLine());  Console.Write("c=");  c = int.Parse(Console.ReadLine());  int max1 = a;  if (-b > max1)  max1 = -b;  if (3 > max1)  max1 = 3;  int max2 = a;  if (2\*b > max2)  max2 = 2\*b;  if (c > max1)  max2 = c;  int max3 = -a;  if (b > max3)  max3 = b;  if (7 > max3)  max3 = 7;  int S = max1 \* max2 - max3;  Console.WriteLine("S={0}",S);  Console.ReadKey();  }  }  } | using System;  using System.Text;  namespace ConsoleApplication3  {  class Program  {  **static int Max(int c1, int c2, int c3)**  **{ int m = c1;**  **if (c2 > m)**  **m = c2;**  **if (c3 > m)**  **m = c3;**  **return m;**  **}**  static void Main(string[] args)  {  int a,b,c;  Console.Write("a=");  a = int.Parse(Console.ReadLine());  Console.Write("b=");  b = int.Parse(Console.ReadLine());  Console.Write("c=");  c = int.Parse(Console.ReadLine());  int max1 = Max(a,-b,3);  int max2 = Max(a, 2\*b, c);  int max3 = Max(-a, b, 7);  int S = max1 \* max2 - max3;  Console.WriteLine("S={0}",S);  Console.ReadKey();  }  }  } |

Якщо метод не повертає якого-небудь значення то його типом є void.

**Формальні параметри методів**

Формальні параметри в списку формальних параметрів розділяються комою. В якості модифікаторів аргументів можуть використовуватися наступні ключові слова

* ref
* out
* params

Описання метода створює окрему область дії імен для його аргументів і локальних змінних, які визначені в блоці, який утворює тіло методу.

При виклику методу створюється окрема копія аргументів і локальних змінних даного методу, після чого відбувається заповнення копій аргументів значеннями, що передаються в списку аргументів. В межах блоку зверненні до аргументів методу здійснюється по іменам, визначеним в списку аргументів.

Є чотири типи формальних параметрів:

* що передаються за значенням (немає ніякого модифікатора)
* передаються за вказівником (вказується модифікатор ref)
* формальні параметри, які використовуються для передачі вихідних значень (вказується модифікатор out)
* формальні параметри змінної довжини (вказується модифікатор params)

Оскільки модифікатор формального параметра є частиною списку формальних параметрів, то в класі можуть бути методи, які відрізняються модифікаторами формальних параметрів.

При передачі аргументу *за значенням* в копію даного аргументу, що створена при виклику методу, просто записується значення, яке передається в даному аргументі. Копія аргументу розглядається як локальна змінна, яка має тип відповідного формального параметра. Зміна значення цієї змінної ніяк не вплине на значення фактичного параметра.

**Приклад**. Створити функцію, що повертає середнє значення трьох дійсних чисел.

class Program

{

**static double Average(double c1, double c2, double c3)**

**{**

**return (c1 + c2 + c3) / 3;**

**}**

static void Main(string[] args)

{ double n, m, k;

Console.WriteLine("n=");

n = double.Parse(Console.ReadLine());

Console.WriteLine("m=");

m = double.Parse(Console.ReadLine());

Console.WriteLine("k=");

k = double.Parse(Console.ReadLine());

double ser= **Average(n, m, k);**

Console.WriteLine("Average = {0} ",ser);

Console.ReadKey();

}

}

*При передачі аргументу за вказівником* в копію даного аргуенту, який має ім’я відповідного формального параметра записується адреса на даний аргумент. Тому, всі зміни стосовно даного формального параметра стосуються також і відповідного фактичного параметра. При описанні відповідного формального параметра та при записі відповідного фактичного параметра використовується модифікатор ref.

Такий спосіб пеередачі параметрів використовується, напиклад, тоді, коли в функції необхідно змінити фактичні параметри.

**Приклад.** Створити функцію, яка більше значення зменшує на 2, а менше збільшує на 3.

class Program

{

**static void Fun1(ref int a, ref int b)**

**{**

**if (a > b)**

**{**

**a -= 2;**

**b += 3;**

**}**

**else**

**{**

**b -= 2;**

**a += 3;**

**}**

**}**

static void Main(string[] args)

{ int n,m;

Console.WriteLine("n=");

n=int.Parse(Console.ReadLine());

Console.WriteLine("m=");

m = int.Parse(Console.ReadLine());

**Fun1(ref n,ref m);**

Console.WriteLine("n= {0} m={1}",n,m);

Console.ReadKey();

}

}

*Аргумент, в який буде записане вихідне значення методу* в основному аналогічний аргументу, який передається за вказівником. Основною відмінністю між ними є те, що для даного аргументу не є обов’язковою ініціалізація фактичного параметра до передачі цього аргументу методу. Але в самому методі фактичний параметр вважається неініціалізованим і має обов’язково бути проініціалізованим до завершення методу. При описанні формального параметра та відповідного фактичного викорстовується модифікатор out.

Такий метод передачі параметрів використовується, наприклад, тоді, коли аргумент є результатом роботи функції.

**Приклад.** Створити функцію, яка повертає максимальне та мінімальне значення з двох дійсних чисел.

class Program

{

static void MinMax(int c1, int c2, out int max, out int min)

{

if (c1 > c2)

{

max = c1;

min = c2;

}

else

{

max = c2;

min = c1;

}

}

static void Main(string[] args)

{ int n,m;

Console.WriteLine("n=");

n=int.Parse(Console.ReadLine());

Console.WriteLine("m=");

m = int.Parse(Console.ReadLine());

int max, min;

**MinMax(n, m, out max, out min);**

Console.WriteLine("max= {0} min={1}",max,min);

Console.ReadKey();

}

}

Отже, зміна значення фактичного параметра в результаті зміни значення формального праметра відбувається тільки у випадку коли передазча здійснюється за вказівником, або як вихідного параметра.

|  |  |  |
| --- | --- | --- |
| **Передача за значенням**  **(немхє ніяких модифікаторів)**  **(інціалізація змінної n обов’язкова)** | **Передача за вказівником**  **(модифіктор ref)**  **(інціалізація змінної n обов’язкова)** | **Аргумент як вихідне значення (модифіктор out)**  **(інціалізація змінної n не обов’язкова)** |
| class Program  {  static void Fun1(int m)  {  m = 1;  }  static void Main(string[] args)  {  int n = **25**;  Fun1(n);  Console.WriteLine("n= {0}",n);  Console.ReadKey();  }  } | class Program  {  static void Fun1(**ref** int m)  {  m = 1;  }  static void Main(string[] args)  {  int n = **25**;  Fun1(**ref n**);  Console.WriteLine("n= {0}",n);  Console.ReadKey();  }  } | class Program  {  static void Fun1(**out** int m)  {  m = 1;  }  static void Main(string[] args)  {  int n;  Fun1(**out n**);  Console.WriteLine("n= {0}",n);  Console.ReadKey();  }  } |
| **Вивід програми:**  n=25 | Вивід програми:  n=1 | Вивід програми:  n=1 |
| **Змінні в пам’яті ЕОМ:**  До виклику:    **Під час роботи функції:**  **(для змінної m виділяється нова пам’ять)**    **Після роботи функції:** | **Змінні в пам’яті ЕОМ:**  До виклику:    **Під час роботи функції:**  **(для змінної m нова пам’ять не виділяється а викорстовується пам’ять змінної n)**    **Після роботи функції:** | **Змінні в пам’яті ЕОМ:**  До виклику:    **Під час роботи функції:**  **(для змінної m нова пам’ять не виділяється а викорстовується пам’ять змінної n)**    **Після роботи функції:** |

Якщо одним із аргументів метода є *одновимірний масив довільного розміру*, то цей аргумент передається останнім і має модифікатор params. Оскільки тип масив не визначено, то це може бути і масив вказівників.

При виклику методів, які мають аргумент з модифікатором params, діють наступні правила:

* при передачі в даному аргументі масивів, елементи якого мають тип, який має явне перетворення до типу елементів вказаного масиву, елементи вхідного масиву передаються по значенням.
* Якщо замість даного аргументу при виклику методу стоїть вираз, тип якого має явне перетворення до типу елемента масиву, то цей аргумент і всі наступні аргументи, тип яких має явне перетворення до типу елементу масива, вважаються елементами вхідного масиву (масиву, що передається). Ці елементи також передаються по значенню. При неможливості перетворення типів генерується помилка. При відсутності аргументу масив має нульову довжину. Для визначення довжини масиву використовується length

Такий спосіб передачі параметрів використовується тоді, коли кількість вхідних аргументів наперед невідома.

class Program

{

static void Max(out double max, params double[] a)

{

if (a.Length > 0)

{

max = a[0];

for (int i = 1; i < a.Length; i++)

{

if (max < a[i])

max = a[i];

}

}

else

{

max = 0;

}

}

static void Main(string[] args)

{

double max;

Max(out max, 0, 1, 2, 3, 4, 5);

Console.WriteLine("max= {0}", max);

Max(out max, 10, 3, 41);

Console.WriteLine("max= {0}", max);

Console.ReadKey();

}

}