Технология ООП

Санкт-Петербургский государственный политехнический университет

06 сентября 2011

Стандартные типы данных в C++

- char — символ;
- int — целое число;
- float — число с плавающей точкой;
- double — число с плавающей точкой двойной точности;
- enum — перечислимый тип задает набор именованных констант целого типа.

Стандартные типы данных в C++

Уточняющие описатели типов: short, long и unsigned делают возможными следующие описания типов:

- unsigned char
- short int
- long int
- unsigned short int
- unsigned int
- unsigned long int

Область действия имен

```
int i=16; // Глобальная переменная
void proc()
{
int i=1;
{
int i=2;
printf ("\n global i=%d, local i=%d" ,::i, i);
}
}
```

Операции языка C++

3 типа операций:

- унарные
- бинарные
- тернарные

Унарные

$- \sim !$	Операции отрицания и дополнения
* &	Разыменование и взятие адреса
sizeof	Определение размера переменной или типа
+	Унарный плюс
++ – –	Приращение (increment) и убывание (decrement)

Бинарные

* / %	Мультипликативные операции
+ −	Аддитивные операции
<< >>	Операции сдвига
< > <= >= == !=	Операции бинарных отношений
& | ^	Побитовые логические операции
&& ||	Логические операции
,	Операции последовательного вычисления

Бинарные

```
float x = -0.1, y = -0.05, z = 0.5;
printf("%x", x<y<z);
```

Операции присвоения

Все эти операции изменяют значение операнда:

$++,\ --,\ +=,\ =,\ *=,\ /=,\ \%=,\ <<=,\ >>=,\ \&=,\ \mid=,\ \hat{}=$

Операции присвоения

Чему будут равны значения переменных i,j в конце фрагмента?

```
int i=0, j=0;
j = i++; //постинкремент
i = ++j; //преинкремент
```

Логические операции

Для логических операций в языке C используются следующие символы:

- || — логическое ИЛИ (OR);
- && — логическое И (AND);
- ! — логическое НЕ (NOT);

Логические операции

```
if (min < n && n < max) // Если n лежит внутри
    диапазона
DoSomething();
```

Побитовые операции

Операции << и >> выполняют сдвиг влево и вправо на указанное во втором операнде число разрядов.

```
int i=64;
i >>= 4;
```

Побитовые операции

- Сдвиг влево переменной устанавливает правые освобождающиеся биты в ноль.
- Сдвиг вправо устанавливает левые биты либо в ноль, либо в единицу в зависимости от типа (беззнаковый — в ноль, знаковый — копией знакового разряда).

Побитовые операции

Операции &, |, ^ являются поразрядными, побитовыми (bitwise) и носят названия: И, ИЛИ, исключающее ИЛИ (XOR).

Побитовые операции

Операция & часто используется для снятия определенных битов в переменной, а операция | — для установки.

```
var |= (1 << i); //установить i-й бит
var &= ~(1 << i); //очистить i-й бит
```

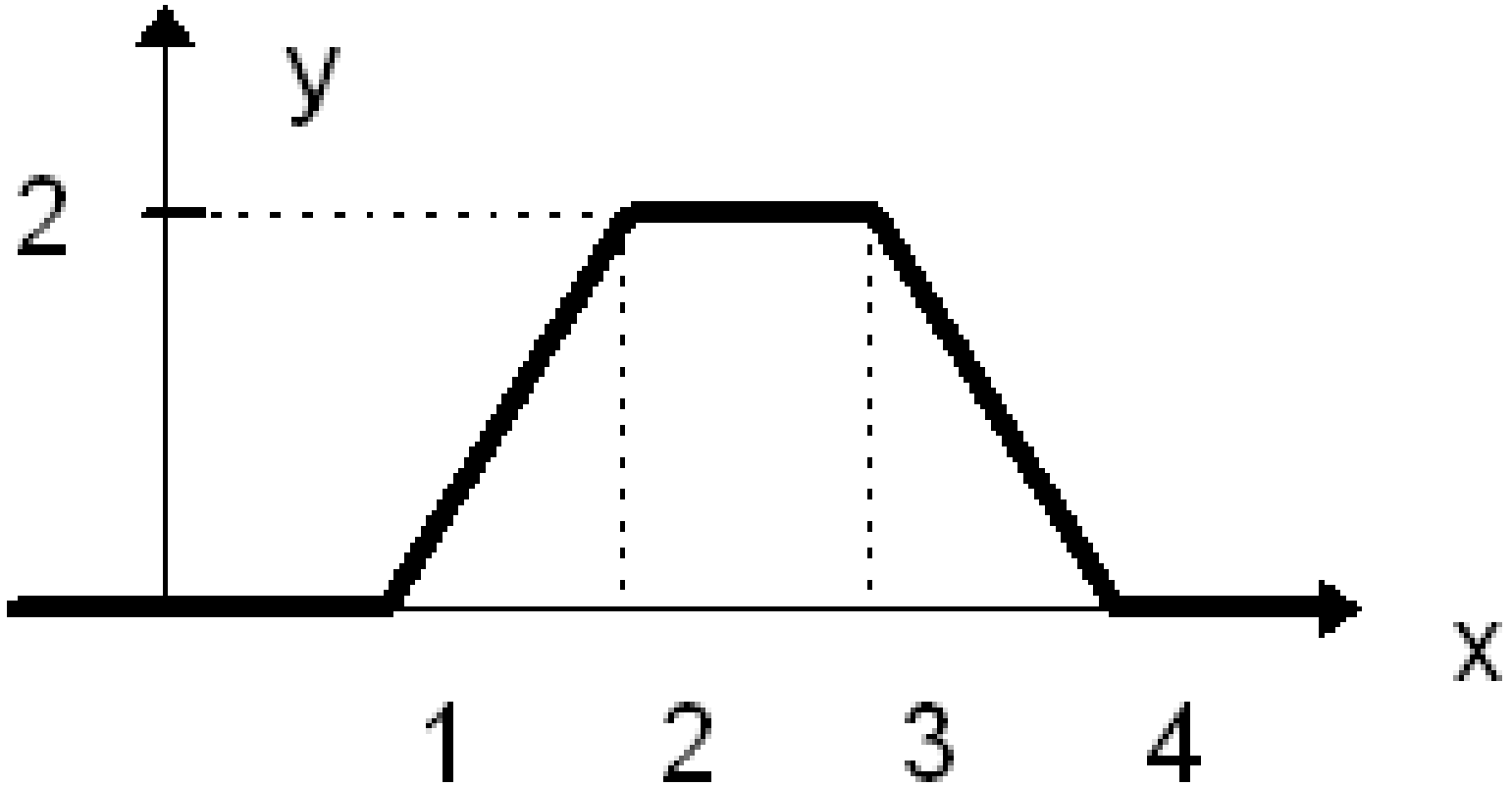
Условные операторы

В языках C и C++ существует две разновидности условных операторов или операторов разветвления: if-statement, и switch-statement

```
if (условие)
{
операторы
}
else
{
операторы
}
```

Условные операторы

Пусть функция задана в виде графика.



Условные операторы

Чтобы задать ее с помощью цепочки вложенных условных операторов, определим переменные double x, y.

```
if (x>1)
  if (x>2)
    if (x>3)
      if (x>4)
        y=0;
      else
        y=-2*x+8;
    else
      y=2;
  else
    y=2*x-2;
else
    y=0;
```

Оператор switch

Алгоритм функционирования оператора можно описать в виде схемы:

```
switch (выражение)
{
case константа1: операторы;
case константа2: операторы;
. . . . .
default: операторы;
}
```

Оператор switch

После выполнения одной из ветвей case управление передается в слудующую ветвь, поэтому для выхода из switch необходимо использовать один из слудующих операторов:

- break;
- goto <метка>;
- continue;
- return;

Тернарные операции

Тернарная операция имеет следующий вид:

```
(условие) ? expr1 : expr2;
```

Тернарные операции

```
i = x < 0 ? -1 : 1;
// Равносильно if (x<0) i=-1; else i=1;
```

Тернарные операции

Специфика тернарной операции заключается в том, что она возвращает значение, т.ẽ. она может быть rvalue (right-hand-side value), что позволяет создавать такие конструкции:

```
i = x < 0 ? -1 : x > 0 ? 1 : 0;
```

Операторы циклов

Три оператора цикла языков C и C++ могут быть описаны следующей схемой:

```
while (условие) {тело цикла}
do {тело цикла} while (условие);
for ( ; ; ) {тело цикла}
```

Операторы циклов

Сколько раз выполнится каждое тело цикла?

```
while(0) {тело цикла}
while (1) {тело цикла}
do {тело цикла} while(0);
```

Операторы циклов

```
int i=1, j=1024;
while (i < j) i=2*i;

int i=1, j=1024;
while (i < j) j=2*i;
```

Цикл for

Последовательность выполнения операторов при реализации цикла:

1. Операторы первой части заголовка (выполняются один раз до входа в цикл).
2. Вторая часть (проверка условия и выход из цикла в случае его нарушения).
3. Операторы тела цикла.
4. Операторы третьей части заголовка.
5. Переход к пункту 2.

Цикл for

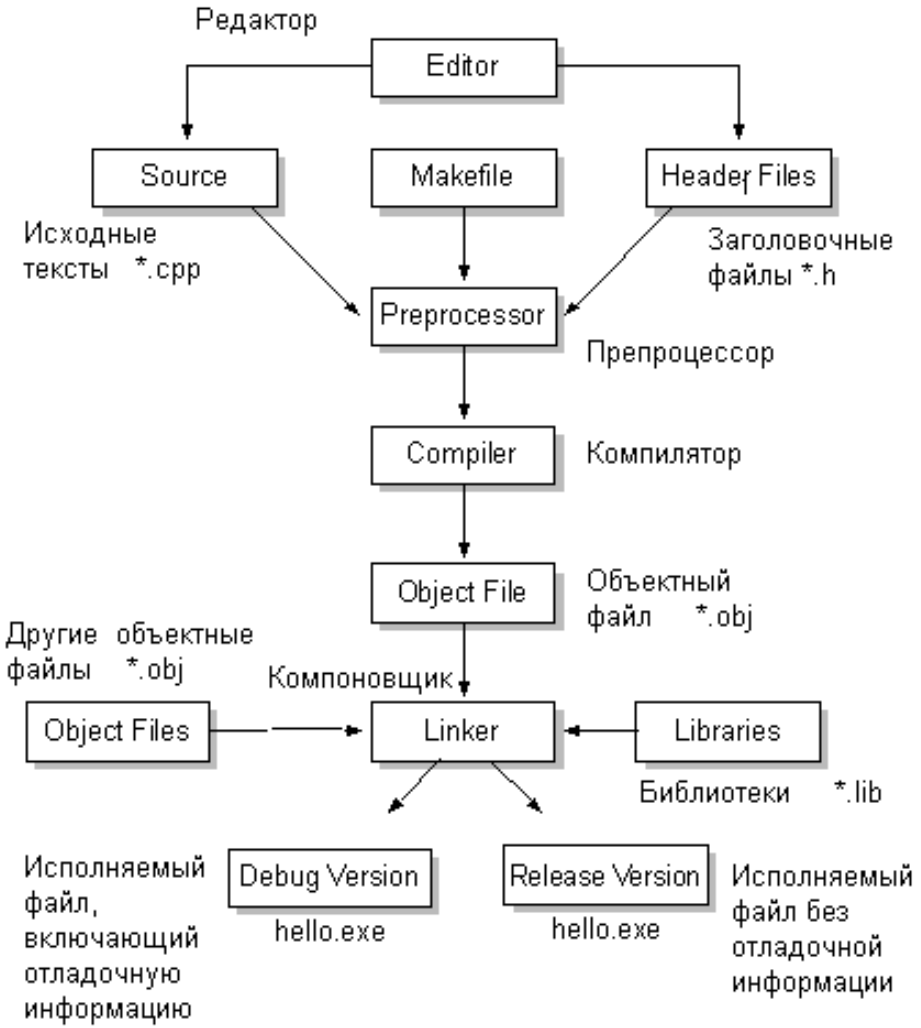
```
char sText[1024];
for (int i=0; i<1024; i++)
sText[i] ' ';  // Заполнение строки текста пробелами
```

Задача

Исправить 1 символ чтобы программа напечатала 20 *. Именно исправить, а не удалить или добавить, пробел тоже символ. Найти 3 решения.

```
int␣main()␣{␣int␣i,␣N=20;␣for␣(i␣=␣0␣;␣i␣<␣N␣;␣i--)
    ␣{␣printf("*");␣}␣}
```

Последовательность этапов генерации исполняемого кода



Указатели и массивы

Имя массива в языке C фактически является указателем на первый его элемент, который соответствует нулевому значению индекса. Если объявлен массив float a[16];, то справедливо равенство a==&a[0].

```
float a[16],*p; // Объявление массива и указателя
for (int i=0; i<16; i++)
a[i] = float(i*i); // Заполнение массива
p = &a[6]; // p указывает на a[6]
*p = 3.14f; // Равносильно a[6] = 3.14;
p++; // Теперь p указывает на a[7]. Произошел сдвиг
     на 4 байта
a[1] = *(p+3); // Равносильно a[1] = a[10]; В a[1]
    попадает 100.
a[2] = ++*p; // Равносильно a[2] = ++a[7]; В a[2]
    попадает 50.
a[3] = *++p; // Равносильно a[3] = a[8]; В a[3]
    попадает 64.
```

Динамическое выделение памяти

Выделим динамически память под матрицу размерностью NxN.

```
// Динамическое размещение массива указателей на
    массивы типа double (строки матрицы)
double **a = new double*[n];
// Динамическое размещение массивов строк матрицы (
    переменные типа double)
for (int i=0; i<n; i++)
a[i] = new double[n];
```

Динамическое выделение памяти

```
// Освобождение памяти, занятой строками матрицы
for (i=0; i<n; i++)
delete [] a[i];
// Освобождение памяти, занятой массивом указателей
    на строки матриц
delete [] a;
```