Программирование и алгоритмизация

Санкт-Петербургский государственный политехнический университет

24 сентября 2014

Размещение переменных

Основные способы:

- статическая память;
- автоматическая память;
- регистровая память;
- динамическая память;

Зависит от контекста определения переменной и использования дополнительных ключевых слов.

Статическая память

Память в которую компилятор помещает объект на все время выполнения программы.
В статической памяти содержатся:

- глобальные переменные;
- переменные из пространств имен;
- статические переменные;

Переменные существуют во время всего выполнения программы.

Автоматическая память

Память в которой по умолчанию располагаются переменные, определенные внутри любого блока и параметры функций. Для таких переменных компилятор отводит место в стеке. Такая память выделяется и освобождается автоматически. Переменные существуют до завершения области видимости блока.

Динамическая память

Память которую явно запрашивает программист, для размещения переменных по мере необходимости.
Такую память программист долэен выделять и освобождать вручную, когда динамический объект становится не нужен.
Такая память также называется кучей (heap).
Переменные существуют до момента вызова программистом оператора удаления.

Правила неявной инициализации переменных

- Статические — нулем.
- Автоматические — не инициализируются, имеют случайное значение области стека где выделена переменная.
- Динамические — не инициализируются, случайное значение выделенной области в куче.

Ключевое слово static

Спецификатор static, требует от компилятора разместить переменную в статической (постоянной) области памяти, а также проинициализировать переменную только один раз.

```
for(int i = 0; i < 3; i++)
{
 int n1 = 1;
 static int n2 = 33;
 static int n3 = n2 * n1;
 n1++;
 n2++;
 n3++;
//i = 0: n1 = ?, n2 = ?, n3 = ?
//i = 1: n1 = ?, n2 = ?, n3 = ?
//i = 2: n1 = ?, n2 = ?, n3 = ?
}
```

Ключевое слово const

Означает, что инициализирующее значение известно на этапе компиляции и не может быть изменено в процессе выполнения программы.
Компилятор Си, резервирует память под переменную с ключевым словом const, но повзволяет использовать ее только для чтения.
Компилятор C++, в большинстве случаев не будет отводить память под переменную, а просто будет подставлять значение.

```
const int N = 1;
N = 5; //???
```

Приведение типов переменных. Неявные приведения типа

Не стоит смешивать в операторах сравнения знаковые и беззнаковые операнды, а также операнды различных типов:

```
int x = -1;
unsigned int y = 1;
if(x < y) {} //???
```

```
char c1 = 0xff;
unsigned char c2 = 0xff;
if(c1 == c2) {} //???
```

```
int n = 0xff;
char c = 0xff;
if(c == n) {} //???
```

Явные приведения типа

При явном приведении типа ответственность за результат ложится на программиста.

```
int x = 1, y = 2;
double z = (double)x/y;
```

Преобразование в стиле Си.

Преобразование типов в стиле C++

- static_cast <тип> (выражение) — преобразование с проверкой типов;
- reinterpret_cast <тип> (выражение) — преобразование без проверки типов;
- const_cast <тип> (выражение) — константное преобразование;

```
int x = 1, y = 2;
double z1 = static_cast<double>(x)/y; //???
double z2 = static_cast<double>(x/y); //???

char c1 = -1;
int y = x; //???
int z = static_cast<unsigned char>(x); //???
```

Преобразование в новом стиле (C++).

Побитовые операции

Операции &, |, ^ являются поразрядными, побитовыми (bitwise) и носят названия: И, ИЛИ, исключающее ИЛИ (XOR). Также существует побитовая операция ~ которая является побитовым отрицанием — все биты изменяют значение на противоположное.

Все побитовые операции применяются, только к целым типам: char, short, int, long, long long. Результатом их работы, также являются целые типы.

Оператор отрицания

```
int x = 1;
int a = -x;
int b = ~x; //???
```

Оператор |

```
char x = 0x1f;
char y = 0xf1;
char z = x | y; //???
```

Оператор &

```
char x = 0x1f;
char y = 0xf1;
char z = x & y; //???
```

Оператор ^

```
char x = 0x1f;
char y = 0xf1;
char z = x ^ y; //???
```

Оператор сдвига

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

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

- 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

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
```

Задание: Поменяйте в 2-х байтной переменной, 2 байта местами. short int x = 0xaabb; //чтобы получилось =>0xbbaa