Основы программирования на C

Наумов Д.А., доц. каф. КТ

Операционные системы и системное программное обеспечение, 2019

Содержание лекции

Краткая характеристика языка

Язык Си – язык программирования высокого уровня, разработанный Д.Ритчи и его коллегами в 70х гг прошлого века. Он представляет собой минималистичный процедурный язык программирования, в котором упор сделан на эффективность компиляции.

- многие возможности вынесены из базовых средств языка и предоставляются стандартной библиотекой, при этом прототипы и семантика функций стандартной библиотеки фиксированы и не зависят от конкретного компилятора, вместе с которым она поставляется;
- Си часто называют «высокоуровневым ассемблером», поскольку он предоставляет возможность писать эффективные программы с весьма низким уровнем абстракции;
- Си не привязан к конкретной аппаратуре или операционной системе и предназначен для написания машинно-независимых и легко переносимых программ.

Базовые типы даных

- Базовые типы: char, int, float и double;
- Квалификаторы: short, long, signed и unsigned.
- Конкретные размеры типов определяются реализацией компилятора.

Переменные и константы

- Имена переменных могут состоять из букв (знак подчеркивания считается буквой) и цифр. Первая литера обязательно буква;
- Большие и маленькие буквы различаются.
- Для внутренних имен значимыми являются первые 31 литера, а для внешних – 6 литер.
- Все переменные должны быть описаны до использования.

```
int a;
long b,c,d;
unsigned char s;
double m[100]; /* m - массив из 100 элементов
                  типа double, индексы принимают
                  значения от 0 до 99 */
```

Константы

- квалификатор const - значение переменной не будет изменяться;
- целые константы: 123, 017, 0x1a, 0X1A, 123L, 1024U
- вещественные константы (double): 123.4, 1.234e2, 123.4L
- вещественные константы (float): 123.4f, 1.234e2f
- символьные константы: '*', 's', '\n', '\t', '\123', '\x12'
- строковые константы: "asdf"неявно заканчивается '\0'

```
const char warnmsg[] = "warning: ";
```

Перечислимый тип

```
enum palette{white, black=10, red, green=-1,
   blue};
/* white, black, red, green и blue - константы
перечислимого типа palette */
enum palette color;
/*color - переменная типа palette, ей может быть
присвоена любая из констант этого типа*/
color = white;
```

- именование констант перечислимого типа уникально в пределах области видимости;
- константы ассоциированы с целым типом Си и могут использоваться везде, где используются константы типа int;
- по умолчанию, константам перечислимого типа присваиваются последовательные значения (0,1,2, ...).

Операции над базовыми типами

Арифметические операции (+, -, *, /, %)

- унарные операции + и - (изменяет знак стоящей справа величины) имеют более высокий приоритет, чем бинарные операции + и - ;
- бинарные операции +(сложение) и −(вычитание) имеют одинаковый приоритет, он ниже приоритета операций *(умножение), /(деление) и %(остаток от деления);
- операция / с операндами целых типов выполняется с усечением дробной части;
- операция % определена только для операндов целых типов;
- арифметические операции с одинаковым приоритетом выполняются слева направо

Операции над базовыми типами

Логические операции (!, &&, ||)

- результатом унарной операции !(логическое НЕ) будет 0 для ненулевого операнда и 1 для нуля;
- приоритет бинарной операции &&(логическое И) выше приоритета бинарной операции ||(логическое ИЛИ).

Операции над базовыми типами

Операции отношения(<, <=, >, >=, ==, !=)

- операции <, <=, >, >= имеют одинаковый приоритет, он ниже приоритета операций сравнения на равенство == и != ;
- операции отношения имеют более низкий приоритет, чем арифметические операции, но более высокий, чем логические операции && и ||.

Операции над базовыми типами

Инкрементная и декрементная операции

- могут быть как префиксными, так и постфиксными, так, например, ++x увеличивает x на 1 до того, как его значение будет использовано, а x++ - после;
- эти операции можно применять только к переменным: запись ++(x+y) не верна.

Операции над базовыми типами

Условная(тернарная) операция (?:)

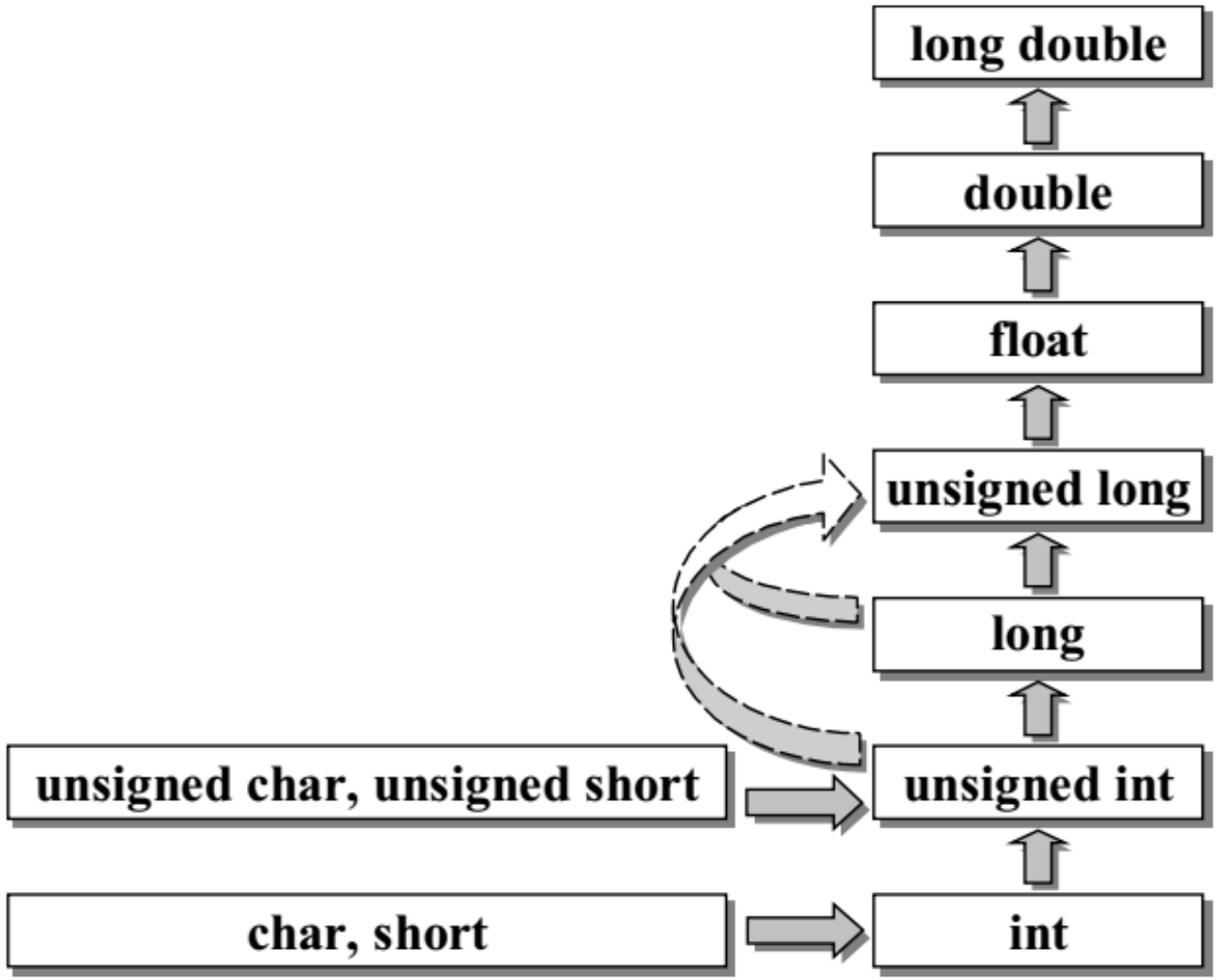
- в выражении x?y:z первым вычисляется значение выражения x, и если оно не равно нулю, то результатом всей операции будет значение выражения y, иначе – значение выражения z.

Операции над базовыми типами

Побитовые операции & , |, <<, >>, ∼, ∧

- все побитовые операции можно применять только к операндам целых типов (знаковым и беззнаковым char, short, int и long);
- результат поразрядных операций & и | не совпадает с результатом логических операций && и ||, например, для x=2 и y=5 результатом x& y будет 0, тогда как результатом x&& y будет 1;
- операции <<(сдвиг влево) и >>(сдвиг вправо) могут использоваться как эффективное умножение и деление на степени 2

Приведение типов



Приведение типов

Пример 1

```
...
int x;
double y=2.9;
x=2+y;
```

Пример 2

```
 ...
int c,d;
c=getchar();

d=c>='0' && c<='9';
```

Операторы управления

Программа на языке Си состоит из набора функций.

- При запуске программы на выполнение управление передается на точку входа в функцию с названием main.
- Функции состоят из операторов.
- Выражение становится оператором, если за ним идет точка с запятой.
- Точка с запятой заканчивает любой оператор, кроме составного оператора

Операторы

- присваивания
- условный оператор (оператор if)
- оператор выбора (переключатель switch)
- оператор цикла с предусловием (while)
- оператор цикла с постусловием (do-while)
- оператор цикла с параметром (for)

Пример 3. Из стандартного входного потока ввести строку и записать ее в массив str. Длина строки не превышает 80 символов.

```
int c,i;
 char str[80];
 i=0;
 while((c=getchar())!='\n') str[i++]=c;
 str[i]='\0';
```

Пример 4. Обнуление элементов одномерного массива.

```
...
int m[10], i;
for(i=0;i<10;i++) m[i]=0;
```

```
...
int m[10], i;
for(i=0;i<10;m[i++]=0);
```

Пример 5. Определение длины строки.

```
...
char s[100];
int len, i;
for(i=0;s[i];i++);
len=i;
```

Пример 6.

```
...
char s[100];

int i, j;
for(i=0,j=0;s[i];i++)
  if(s[i]!=' ') s[j++]=s[i];
s[j]='\0';
```

Пример 7.

```
#include <stdio.h>
main()
{   int c, m[256]={0}, i, j, max;
    while((c=getchar())!='\n') m[c]++;
    for(i=1,max=m[0];i<256;i++)
      if(max<m[i]) max=m[i];
    for(i=0;i<256;i++)
      if(m[i]){
        printf("%c",i);
        for(j=0;j<80*m[i]/max;j++) putchar('*');
        putchar('\n');
      }
}
```