Базовые понятия языка Си

Рассматриваемый материал

1. **Адреса и указатели.**
2. **Операции над указателями.**
3. **Применение указателей.**

Адреса и указатели

Указатель – это переменная, значением которой является *адрес* объекта конкретного типа.

Если в некоторый момент времени адрес объекта неизвестен или переменная – указатель еще не получила значения, то такой переменной придается значение специальной константы NULL, не равное никакому адресу («нулевой» адрес).

Указатели, как и любые переменные программы, должны быть объявлены. В объявление указателя входит указание типа объекта, на который ссылается указатель, признак (символ *) того, что переменная является указателем, имя указателя.

Адреса и указатели

Примеры объявления указателей:

```
int *f; /*указатель на объект целого типа*/

float *a,*b; /*указатели на объекты вещественного типа*/
```

1.Обычный указатель (кроме указателя типа void) может ссылаться на объекты только того типа, который указан при объявлении.

2. После объявления значение указателя не определено (соответствует NULL).

Получение адресной информации осуществляется с помощью унарной операции &. Например, адрес переменной "A" формируется на основе выражения

```
&A
```

Это выражение в ходе выполнения программы позволяет определить адрес участка памяти, выделенного переменной A.

Операции над указателями

1. Присваивание;
2. Разыменование;
3. Унарные операции сложения и вычитания;
4. Аддитивные операции;
5. Операции сравнения;
6. Получение адреса самого указателя.

Операция присваивания

Предполагает, что справа от знака присваивания стоит:

1. Другой указатель, который имеет конкретное значение;
2. Константа NULL;
3. Адрес объекта того же типа, что и указатель слева.

Примеры записи операции присваивания:

```
c=d;
```

Предполагается, что переменные c и d являются указателями одного типа. После этой операции оба указателя обеспечивают доступ к одному участку памяти.

```
c=NULL;
```

```
c=&a; //Указателю присваивается значение адреса
```

Если указателю одного типа требуется присвоить значение указателя другого типа, то применяют приведение типов:

```
int *k;
```

```
float *z;
```

```
k=(int) z; //преобразование перед присваиванием
```

Операция разыменования (обращения по адресу, раскрытия ссылки)

1. Позволяет обратиться к соответствующему объекту (переменной). Эта операция является унарной.
2. Операция разыменования обозначается символом звездочки. Выражение *f соответствует объекту, на который указывает указатель "f".
3. Выражение *f имеет все "права" переменной, например оператор *f=0 означает занесение нуля в тот участок памяти, адрес которого хранится в переменной – указателе "f".
4. Разыменовывание указателя, соответствующего NULL, недопустимо.

Унарные операции сложения ++ и вычитания – –

Позволяют увеличить или уменьшить значение указателя на величину, равную длине участка памяти, занимаемого соответствующим типом данных. Тем самым обеспечивается переход к началу участка памяти для соседней переменной.

В частности, переменные типа char занимают в памяти участок длиной в 1 байт, переменные типа int – два байта, переменные типа long int и float – 4 байта, переменные double – 8 байт, long double – 10 байт.

Аддитивные операции

В полном объеме эти операции не разрешены. Переменные типа указатель **нельзя суммировать**, но к переменной типа указатель **можно добавлять целую величину k**. При этом указатель увеличивается на величину k*r, где r – длина участка в байтах, занимаемого соответствующей переменной. Так при добавлении к указателю переменной типа float величины, равной трем, указатель увеличится на значение 3*4 = 12 байт.

Аналогично изменяется значение указателя при *вычитании из его значения целой величины*.

Операция *вычитания* применима к указателям одного типа. Из значения одного указателя можно вычитать значение другого указателя. Результатом операции является целая величина со знаком. Значение результата формируется в "масштабе" соответствующего типа данных. Например, при вычитании указателей двух соседних величин типа int результат будет равнее единице, а не двум, хотя переменные этого типа занимают два байта.

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

Применимы только при сопоставлении указателей одного типа или при сравнении указателя с константой NULL.

Операция получения адреса

Позволяет получить адрес указателя. Указатель является переменной, поэтому и к нему применима операция получения адреса. Например, следующий фрагмент программы позволит отобразить значение адреса указателя

```
float *a;

printf("\n адрес указателя %p", &a);
```

Отображение указателей и адресов производится с использование спецификации **p** – представление шестнадцатеричных чисел.

Отображение результатов вычитания указателей осуществляется с использованием спецификации **d** для целых чисел.

Применение указателей при работе с массивами

В языке Си при традиционном объявлении массива его имя становится указателем на начало области памяти, выделенной под массив, т.е. имя массива без индексов является указателем на первый элемент, т.е. элемент с нулевым индексом этого массива.

В частности:

```
float r[10], *pn1;

pn1=r;

pn1=&r[0];
```

В этом фрагменте программы два последних оператора эквиваленты.

Применение указателей при работе с массивами

Элементы массива в памяти располагаются последовательно друг за другом, поэтому, изменяя указатель на величину, равную длине элемента массива, можно переходить к соседним элементам.

Имя массива интерпретируется как указатель – константа, следовательно, к нему не применимы операции изменения значений, оно не может стоять слева от знака присваивания.

Фрагмент программы для подсчета среднего значения элементов массива с применением указателей

```
float r[10],s, *pn1;

for (pn1=&r[0], s=0; pn1<=&r[9]; s+=*pn1, pn1++);

s/=10;
```

Применение указателей при работе с массивами

В приведенном примере тело цикла отсутствует (за оператором for стоит символ ";"), все действия по подсчету суммы записаны в заголовке цикла.

Выражения:

```
pn1=&r[0]
```

обеспечивает установку указателя на первый (т.е. нулевой) элемент массива;

```
s+=*pn1
```

суммирование элементов массива;

```
pn1++
```

изменение указателя для перехода к следующему элементу массива (фактическое приращение указателя происходит не на единицу, а на 4, так как каждый элемент в памяти занимает 4 байта).

Операция индексирования r[i] эквивалентна *(r+i), где r – имя массива, i – выражение целого типа. Для многомерных массивов правило остается такое же. Например, z22[n] [m] [k] эквивалентно *(z22[n] [m] +k) и далее *(*(*(z22+n)+m)+k).

Применение указателей в параметрах функции

Схема вызова функции с помощью передачи параметров по значению не позволяет непосредственно возвращать из функции несколько результатов.

Существует косвенная возможность изменения значений переменных в вызывающей программе на основе применения указателей:

1. В качестве *фактических параметров* при обращении к функции задаются *адреса объектов* вызывающей программы;

2. В *функции* с помощью *разыменования указателей* обеспечивается доступ к соответствующему объекту вызывающей программы, в том числе, это позволяет *изменять* значение объекта программы.