Лекция 10

АДРЕСАЦИЯ ПЕРЕМЕННЫХ И УКАЗАТЕЛИ В СРЕДЕ VISUAL C++

Цель лекции. Изучить адресацию памяти и исследовать особенности применения указателей в Visual C++ 2010.

Вопросы лекции:

1. Понятие адреса переменной в Visual C++.
2. Понятие указателя в языке в Visual C++.
3. Разыменование указателей в Visual C++.
4. Операции с указателями в Visual C++.
5. Указатели и массивы в Visual C++.

1. Понятие адреса переменной в Visual C++ 2010

Переменная занимает в памяти компьютера определенную область (набор ячеек). Расположение переменной в памяти, т. е. данный набор ячеек, определяется адресом. При объявлении переменной для нее резервируется место в памяти. Размер зарезервированной памяти зависит от типа данной переменной.

Для доступа к содержимому выделенной памяти служит его **имя** (идентификатор). Для того чтобы узнать адрес конкретной переменной, применяется операция **взятия адреса**. Синтаксис операции следующий:

&ИмяПеременной,

т. е. перед именем переменной ставится знак **&**.

Рассмотрим программу, в которой используется операция взятия адреса для двух переменных **A** и **B**:

Пример 1. Программа, в которой вычисляются адреса двух переменных **A** и **B**.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
        double A=23.1;
        double B=57.88;
        cout<<"Znachenie A= "<<A<<endl;
        cout<<"Adres A= "<<&A<<endl;
        cout<<"Znachenie B= "<<B<<endl;
        cout<<"Adres B= "<<&B<<endl;
getch();
return 0;
}
```

Результат выполнения программы следующий:

```
Znachenie A= 23.1
Adres A= 002EF800
Znachenie B= 57.88
Adres B= 002EF7F0
```

При исследовании программ по этой теме следует учитывать, что адреса переменных будут отличаться для различных компьютеров.

Адреса переменных записаны в шестнадцатиричной системе счисления.

Адреса локальных переменных размещаются в стеке. Поэтому их адреса следуют в обратном порядке (стек растет в направлении младших адресов). Разница в адресах **A** и **B** всегда будет одинаковая и при 4-х байтовом представлении чисел типа **int** составит 4 байта.

2. Понятие указателя в Visual C++ 2010

В языке C++ есть возможность осуществлять непосредственный доступ к памяти. Для этого предусмотрен специальный тип переменных – указатели.

Указатель – это переменная, содержащая адрес некоторого объекта. Объектом может быть переменная или функция. В общем случае указатель – это целое число.

На рис. 1 показана взаимосвязь между адресом переменной и указателем на этот адрес.

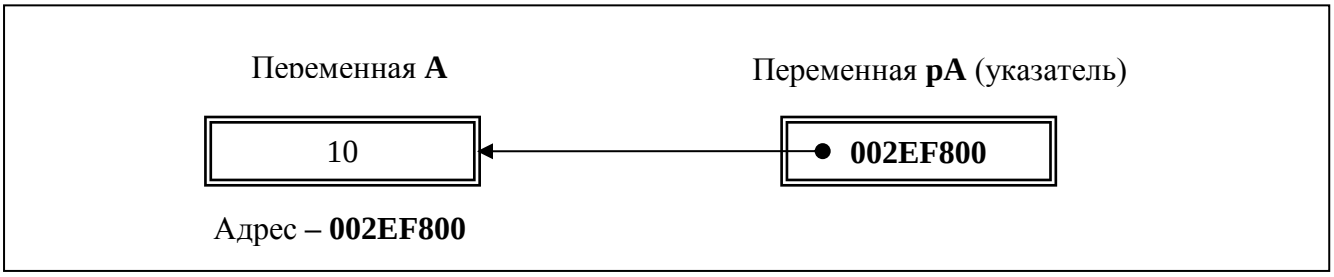

Рис. 1. Взаимосвязь между адресом переменной и указателем на этот адрес.

Если переменная будет указателем, то она должна быть объявлена в программе. Указатель в программе объявляется следующим образом:

ТипОбъекта *Идентификатор;

Здесь "**ТипОбъекта**" определяет тип данных, на которые ссылается указатель с именем "**Идентификатор**". Символ * (звездочка) означает, что следующая за ней переменная является указателем. При объявлении указателя под него резервируется 4 байта.

Примеры объявления указателей:

```
Char *A
int *temp, i, *z;
double f, *ptr;
```

Здесь объявлены указатели **ch**, **temp**, **z**, **ptr** и переменные **i** и **f** .

Поскольку указатель является ссылкой на некоторую область памяти, ему может быть присвоен только значение **адреса переменной**, а не значение самой переменной.

Рассмотрим пример объявления и инициализации указателя.

Пример 2. Программа, в которой инициализируются указатели **uA** и **uB** переменных **A** и **B** с присвоением им значений адресов этих переменных.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
  double A=57.97;
  double *uA=&A;                          //Инициализация указателя uA и присваивание ему адреса A
  double B=340;
  double *uB;                             //Инициализация указателя uB
  uB =&B;                                 //Присваивание указателю uB значения адреса B
  cout<<"A= "<<A<<"   &A= "<<&A<<endl;    //Вывод переменной A и адреса A
  cout<<"    uA = "<<uA<<endl;            //Вывод указателя uA
  cout<<"B= "<<B<<"   &B= "<<&B<<endl;    //Вывод переменной B и адреса B
  cout<<"    uB = "<< uB <<endl;          //Вывод указателя uB
  getch();
  return 0;
}
```

Результат выполнения программы следующий:

```
A= 57.97    &A= 001CFB4C
            uA  = 001CFB4C
B= 340    &B= 001CFB30
            uB  = 001CFB30
```

Очевидно, что значение указателя совпадает со значением адреса соответствующей переменной.

3. Разыменование указателей.

Указатели помогают осуществлять непосредственный доступ к памяти. Для того чтобы получить (прочитать) значение, записанное по адресу, который находится в указателе, используют операцию непрямого обращения или **разыменования** (*). Для этого используется имя указателя со звездочкой перед ним.

Пример 3. Программа, в которой разыменуются указатели **uA** и **uB**, т. е. определяются значения переменных A и B по значениям указателей на эти переменные.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
        double A=57.97;
        double *uA=&A;                  //Инициализация указателя uA и присваивание ему адреса A
        double B=340;
        double *uB;                             //Объявление указателя uB
        uB =&B;                                 //Присваивание указателю uB значения адреса B
```

```
        cout<<"A= "<<A<<"   &A= "<<&A<<endl;
        cout<<"    uA = "<< uA <<endl;                  //Вывод указателя uA
        cout<<"* uA =  "<<* uA <<endl;                  //Разыменование  указателя uA и вывод результата
        cout<<"B= "<<B<<"   &B= "<<&B<<endl;
        cout<<"    uB = "<< uB <<endl;                  //Вывод указателя uB
        cout<<"* uB = "<<* uB <<endl;                   //Разыменование  указателя uB и вывод результата
        getch();
        return 0;
}
```

После выполнения программы на экран будет выдана следующая информация:

```
A= 57.97   &A= 0026FD68
             uA = 0026FD68
* uA =  57.97
B= 340   &B= 0026FD4C
             uB = 0026FD4C
* uB = 340
```

3. Операции с указателями.

Язык С++ позволяет работать с указателями также, как и с переменными стандартных типов. Однако операции над указателями отличаются некоторыми особенностями.

С указателями можно выполнять следующие операции: разадресация (*), присваивание, сложение с константой, вычитание,инкремент (++), декремент (- -), сравнение, приведение типов. При работе с указателями часто используется операция получения адреса (&).

Операция присваивания.

Указатели одного и того же типа могут использоваться в операциях присваивания, как и другие любые переменные.

Пример 4. Программа, в которой применяется операциях присваивания указателей **px** и **g** , а **g** затем разыменуется для проверки **того**, совпадает ли значение ***g** со значением переменной **x**.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
        int x=10;
        int *px, *g;
        px=&x;
        g=px;
        cout<<"px= "<<px<<endl;
        cout<<"g= "<<g<<endl;
        cout<<"x= "<<x<<endl <<  " *g= "<<*g<<endl;
  getch();
  return 0;
}
```

После выполнения программы на экран будет выдана следующая информация.

```
px= 002FFB74
g= 002FFB74
x= 10
 *g= 10
```

Очевидно, что операция присваивания между указателями и последующее разыменование указателя **g** не внесли погрешностей и значение разыменованного указателя **g** совпадает со значением переменной **x**.

Сравнение указателей.

Указатели можно сравнивать, применяя все 6 операций сравнение.

P1 = = P2 – сравнение на равенство;
P1 != P2 – сравнение на неравенство;
P1 < P2 – меньше;
P1 < = P2 – меньше или равно;
P1 > P2 – больше;
P1 > = P2 – больше или равно.

Сравнение **p < g** , например, означает, что адрес, находящийся в **p**, меньше адреса, находящегося в **g**.

Операция sizeof

Как и к любой переменной или типу данных, к указателям можно применять операцию определения размера – **sizeof**. В современных ПК указатель имеет размер 4 байта (32 двоичных разряда) и может адресовать **2^{32} = 4 Гбайт** памяти.

К указателям можно применять не только операцию **sizeof** но и одноименную функцию. Исследуем этот прием в нижеследующей программе.

Пример 5.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
  {
        unsigned long A=546213;
        bool T=false;
        unsigned long *pA=&A;
        bool *pT=&T;
        cout<<sizeof pA<<endl;
        cout<<sizeof (pA)<<endl;
        cout<<sizeof pT<<endl;
        cout<<sizeof (pT)<<endl;
getch();
return 0;
}
```

Вид экрана после выполнения программы:

```
4
4
4
4
```

4. Указатели и массивы

В языке С++ принято, что имя массива – это адрес ячейки памяти, начиная с которой располагается массив. Другими словами, имя массива – это адрес первого (нулевого) элемента массива.

Если объявлен массив

int A[12];

то **Mas** является указателем на массив, точнее на первый элемент массива.

Таким образом**,**

pA = A[0];

Для того чтобы получить значение 8-го элемента массива **A**, необходимо задать

A[8]=*(A+7) .

В С++ **n**-й элемент массива определяется как

A[n] =*(A[0]+n).

Пример 7. Исследуем программу, которая вводит в память в клавиатуры целочисленный массив, вычисляет сумму его элементов и выводит на экран эту информацию с использованием указателей.

```
#include "stdafx.h"
#include <conio.h>
#include "iostream"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
  {
        const int M=50;
        int i, m, S, Mas[M];
        int *pMas;
        pMas=Mas;
        cout<<"Vvedite razmer massiva m:"<<endl;
        cin>>m;
        cout<<"Vvedite massiv:"<<endl;
        for (i=0; i<m; i++, pMas++)
              cin>>*pMas;
        cout<<endl;
```

```
        S=0;
        pMas=pMas-m;
        for (i=0; i<m; i++, pMas++)
                S=S+*pMas;
        pMas=pMas-m;
        for (i=0; i<m; i++, pMas++)
                cout<<*pMas<<' ';
        cout<<endl<<"S= "<<S<<endl;
getch();
return 0;
}
```

Вид экрана после выполнения программы:

```
Vvedite razmer massiva m:
7
Vvedite massiv:
1 2 3 4 5 6 7

1 2 3 4 5 6 7
S= 28
```

Вывод. В языке С++ есть возможность осуществлять непосредственный доступ к памяти. Для этого предусмотрен специальный тип переменных – указатели.

Указатель – это переменная, содержащая адрес некоторого объекта.

Поскольку указатель является ссылкой на некоторую область памяти, ему может быть присвоен только адрес переменной, а не ее значение.

Указатели помогают осуществлять непосредственный доступ к памяти. Для того чтобы получить (прочитать) значение, записанное по адресу, который находится в указателе, используют операцию непрямого обращения или **разыменования** (*). Для этого используется имя указателя со звездочкой перед ним.

Вопросы для самоконтроля.

1. Что будет выведено на экран после выполнения программы **int A=300; cout<<&A;**?
   1) Значение **A**, то есть 300
   2) Адрес ячеек, в которых записано значение **A**
   3) Сообщение об ошибке
2. Укажите правильное объявление указателя в С++.
   1) ***double pA;**
   2) **double *pA;**
   3) **double pA*;**
3. Возможна ли следующая инициализация указателя: **char A=”yes”; char *pA=&A;**?
   1. Возможна
   2. Невозможна
   3. Такой конструкции в С++ нет
4. Что означает оператор **double *pA=&A;**?
   1. Инициализация переменной **A**
   2. Объявление указателя **pA**
   3. Инициализация указателя **pA** и присваивание ему адреса переменной **A**
5. Что такое указатель в языке С++?
   1. Адрес некоторой переменной или функции
   2. Переменная, содержащая адрес некоторой переменной или функции
   3. Стандартная функция
6. Как объявляется указатель с именем **pA** в языке С++?
   1. **int *pA** 2. **int «pA** 3. **int &A**
7. Как обозначается операция взятия адреса переменной **A** в языке С++?
   1. ***pA** 2. **«pA** 3. **&A**
8. Как обозначается операция разыменования указателя **pA** в языке С++?
   1. ***pA** 2. **«pA** 3. **&pA**
9. Что означает операция разыменования указателя в языке С++?
   1. Получение адреса переменной в указателе
   2. Получение значения переменной, записанной по адресу, который находится в указателе
   3. Запись переменной по адресу, который находится в указателе
10. Какое значение примет **Y** в программе **double X=10.1; double *pA; pA=&X; Y=*pA;**
    1. **Y**=10.1
    2. В переменную **Y** запишется адрес, по которому находится значение **X**
    3. Будет выдано сообщение об ошибке с указанием на последнюю строку

11. Можно ли в языке C++ выполнять арифметические операции над указателями?
  1. Можно
  2. Нельзя
  3. Можно выполнять только операцию присваивания
12. Что будет выведено на экран дисплея после выполнения программы
  int *pA; for (i=0; i<100; i++) cout<<*(pA+i)<<" ";?
  1. Значения элементов какого-то массива
  2. Значения указателей
  3. Такую конструкцию в языке C++ использовать нельзя
13. Укажите на возможность такого объявления указателя **int **ppA;**
  1. Возможно
  2. Невозможно
  3. Все зависит от содержания программы
14. На сколько байтов изменится значение **pC** в программе **int C[20]; int *pC=&C; pC++;**?
  1. Не изменится 2. Увеличится на 4 байта 3. Уменьшится на 4 байта