Міністерство освіти і науки України
Харківський національний автомобільно-дорожній університет
Кафедра інформаційних технологій та мехатроніки

МЕТОДИЧНІ ВКАЗІВКИ

до лабораторних робіт з дисципліни

“Інформаційні технології”

“Програмування на мові С++ у середовищі Microsoft Visual Studio 2010”

для студентів напряму підготовки 6.050702 ” Електромеханіка ”,
галузь знань 0507 ” Електротехніка та електромеханіка ”

Розроблено та надруковано доц. Симбірським Г.Д.

Харків, 2018

Лабораторная работа № 9.

АДРЕСАЦИЯ ПЕРЕМЕННЫХ И УКАЗАТЕЛИ В СРЕДЕ VISUAL C++ 2010

Цель работы: изучение адресации памяти и исследование особенностей применения указателей в Visual C++ 2010.

1. Понятие адреса переменной и указателя в Visual C++ 2010

Переменная занимает в памяти компьютера определенную область (набор ячеек). Расположение переменной в памяти, т. е. данный именованный 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 **&**.

Задание 9.1. Исследовать программу, в которой используется операция взятия адреса для 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;        //Операция взятия адреса переменной
                                    //A и вывод адреса  &A
cout<<"Znachenie B= "<<B<<endl;
cout<<"Adres B= "<<&B<<endl;        //Операция взятия адреса перемен-
                                    //ной B и вывод адреса  &B
getch();
return 0;
}
```

После отладки и выполнения программы получим следующий результат:

```
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 для каждого конкретного компьютера будут отличаться от приведенных в методическом пособии.

Создайте в текстовом процессоре **Word** файл **Результат_Фамилия_Лр9**. Поля документа сделайте по 0,5 см.

Поместите окно DOS с результатами решения **Задания 2.1** в центральной части окна **Microsoft Visual Studio** ниже программного кода **Lr2-1.cpp** (см. рис. 1.2) и нажмите клавишу **<Prt Scr**>, после чего вставьте полученную копию экрана в файл **Результат_Фамилия_Лр2**. Над вставленным рисунком проставьте номер задания – 2-**1**. Файл результатов не закрывайте до получения оценки за выполненную практическую часть работы в тетрадь с отчетом.

Закройте окно DOS, откройте пункт меню **Файл** и выполните команду **Закрыть решение.**

Внимание! Результаты следующих заданий данной лабораторной работы сохраняйте строго в соответствии с приведенным выше порядком действий!

В языке С++ есть возможность осуществлять непосредственный доступ к памяти компьютера. 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. 92.1 показана взаимосвязь между адресом переменной и указателем на этот адрес.

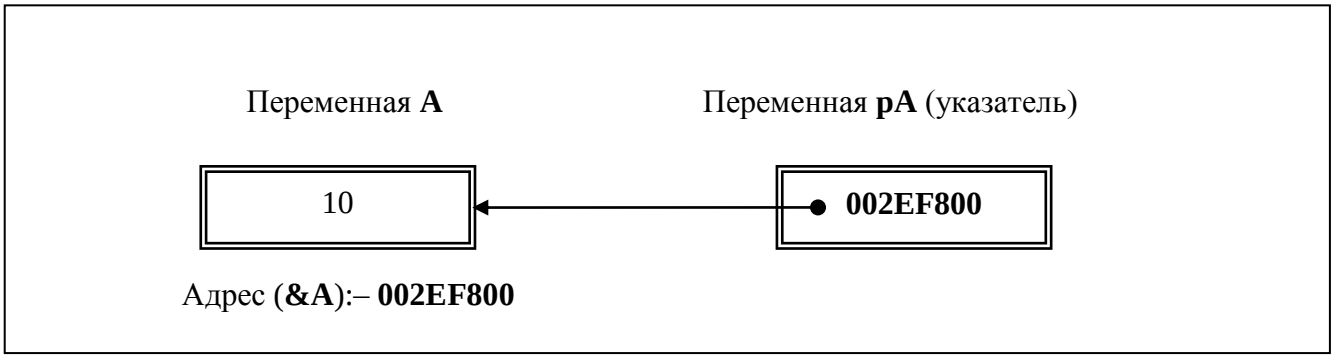


Рис. 9.1. Взаимосвязь между адресом переменной и указателем на этот адрес.

Такая взаимосвязь отображается следующим образом:

pA = &A;

Приведенная инструкция означает, что переменной **pA** присваивается адрес ячейки со значением **A** (в нашем случае **002EF800**).

Операндом оператора **&** не могут быть ни выражение, ни константа, ни регистровая переменная. Унарный оператор **&** называется **оператором адресации.**

Имена указателям даются в соответствии с правилами, приятыми в языке программирования **C** для обычных переменных.

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 **A**, **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.

Задание 9.2. Исследовать программу, в которой объявляются и инициализируются указатели **pA** и **pB** с присваиванием им значений адресов двух переменных **A** и **B**.

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

Результат выполнения программы следующий:

```
A= 57.97   &A= 002CF7B4
           pA  = 002CF7B4
B= 340   &B= 002CF798
           pB  = 002CF798
```

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

2. Разыменование (разадресация) 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.

Тогда для ранее использовавшихся обозначений будет справедливо записать

A=*pA.

Задание 9.3. Исследовать программу, в которой разыменуются указатели **pA** и **pB**, 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 *pA=&A;                  //Инициализация указателя uA и
                        //присваивание ему адреса A
double B=340;
double *pB;                     //Объявление указателя pB
pB =&B;                         //Присваивание указателю pB значения
                        //адреса B
cout<<"A= "<<A<<"   &A= "<<&A<<endl;
cout<<"  pA = "<< pA <<endl;        //Вывод указателя pA
cout<<"*pA =  "<<*pA <<endl;       //Разыменование  указателя pA и вывод
                        //результата
cout<<"B= "<<B<<"   &B= "<<&B<<endl;
cout<<"   pB = "<< pB <<endl;       //Вывод указателя pB
cout<<"*pB = "<<*pB <<endl;        //Разыменование  указателя pB и вывод
                        //результата
getch();
return 0;
}
```

После выполнения программы внимательно изучите на экране следующую информацию:

```
A= 57.97    &A= 0016FE90
              pA  = 0016FE90
* pA =  57.97
B= 340    &B= 0016FE74
              pB  = 0016FE74
* pB = 340
```

Язык программирования C++ позволяет работать с указателями так же, как и с переменными стандартных типов. Однако операции над указателями отличаются некоторыми особенностями.

Задание 9.4. Разработать программу определения адресов целых чисел от 0 до 9 и строчных букв латинского алфавита.

Программный код решения задания следующий:

```
#include <stdio.h>
#include <conio.h>
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int i, j = 0;
  char c='a', *psymbol;
// Адрес символа 'a'
psymbol=&c;
printf("\n\t Figures, symbols and their addresses:\n");
   for (i = 0; i < 10; ++i)
      printf("\n\t %3d) %2d --> %5p", i + 1, i, &i);
printf("\n");
for (*psymbol='a'; *psymbol<='z'; (*psymbol)++)
      printf("\n\t %3d) %2c --> %5p", ++j, *psymbol, psymbol);
```

```
printf("\n\n Press any key: ");
getch();
return 0;
}
```

В программе использован спецификатор формата **%5p** для определения адреса переменных. Число **5** определяет отступ от левого края на пять позиций.

После выполнения программы получим следующий результат:

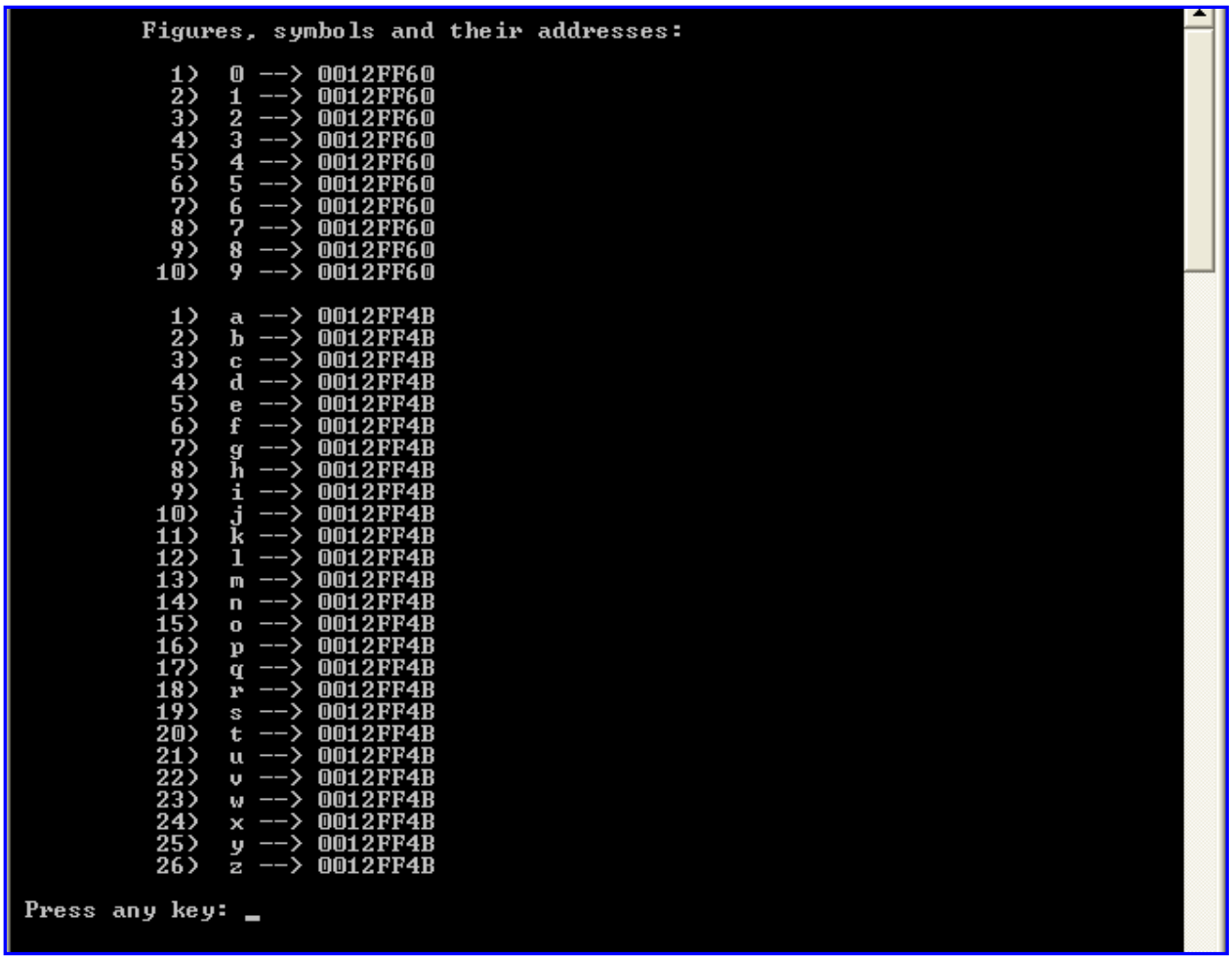


Рис. 9.2. Вывод адресов цифр и строчных букв

Задание 9.5. Внесите в программный код для Задания 2.4 такие изменения:

добавьте определение адресов прописных букв латинского алфавита и выведите их дополнительным столбцом к адресам строчных букв.

3. Операция присваивания указателей

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

Для указателей одного типа можно, например, выполнять присваивание без разыменования, поскольку указатели сами по себе являются переменными.

Пусть определен еще один указатель типа **int**, например **p2A**. Тогда возможно произвести присвоение:

pA = p2A;

После этого указатель **p2A** будет указывать на ту же переменную **A**, что и указатель **pA**.

Задание 9.6. Исследовать программу, в которой применяется операциях присваивания указателей **px** и **g**, а **g** затем разыменуется для проверки - определяется, 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**.

4. Операции с указателями

В языке **C** допустимы следующие основные операции над указателями:

1. Присваивание; получение значения того объекта, на который он указывает (синонимы: косвенная адресация, разыменование, раскрытие ссылки) (см. выше);
2. Получение адреса самого указателя;
3. Унарные операции изменения значения указателя;
4. Аддитивные операции;
5. Операции сравнений (отношений).

Унарные операции **++** и **--** позволяют позиционировать указатель на следующую и предыдущую ячейки памяти, в которых хранятся значения типов, связанных с типом указателя. При этом значение указателя меняется на величину, определяемую размером соответствующего типа. Например, для указателя типа **char*** операция **++** увеличит значение адреса на **sizeof (char)**, для указателя типа **int*** операция **--** уменьшит значение адреса на **sizeof(int)** и т. д. Это свойство унарных операций **++** и **--** используется для последовательного обращения к значениям одного типа, связанного с типом указателя, хранящимся в смежных ячейках памяти. В таком смысле унарные операции **++** и **--** сходны с операциями увеличения и уменьшения счетчика цикла при последовательном обращении к элементам массива.

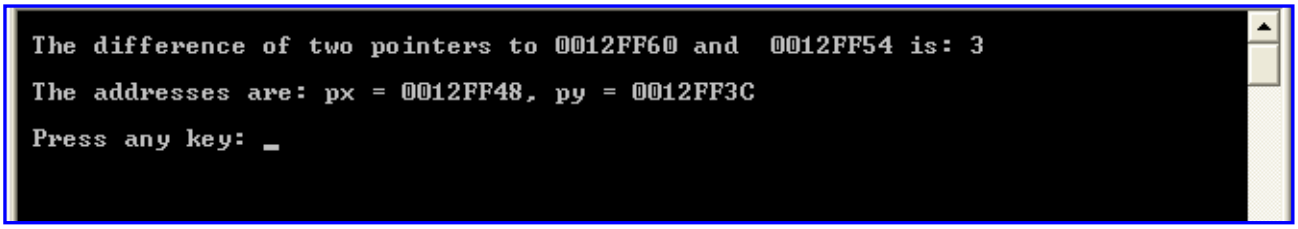
Задание 9.7. Разработать программу однозначного задания типа разностей указателей и определения адресов заданных указателей.

При решении данного примера подключим заголовок **stddef.h** для определения типа разности указателей с помощью зарезервированного имени типа **ptrdiff_t**.

Программный код решения задачи следующий:

```
#include <stdio.h>
#include <conio.h>
#include <stddef.h>
#include "stdafx.h"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int x, y;
int *px, *py;
ptrdiff_t z;
// Взятие адресов переменных
px = &x;
py = &y;
// Разница двух указателей
z = px - py;
printf("\n The difference of two pointers to %p and  %p is: %d",px,py,(int) z);
printf("\n\n The addresses are: px = %p, py = %p\n", &px, &py);
printf("\n Press any key: ");
getch();
return 0;
}
```

После выполнения программы на экран будет выведена следующая информация:



Задание 9.8. В программном коде для задания 2.7:

1. Поменяйте местами переменные **x** и **y**. Проанализируйте результат выполнения программы.
2. Для переменных произведите инициализацию в соответствии с номером компьютера, на котором выполняется лабораторная работа, и текущего дня недели.
3. Рассмотрите решение примера для следующих типов: **char**, **long int**, **unsigned int, float**, **double**, **long double**.
4. Вывод результатов осуществите с помощью одной функции **printf()**.

Задание 9.9. Разработать программу арифметических операций с указателями.

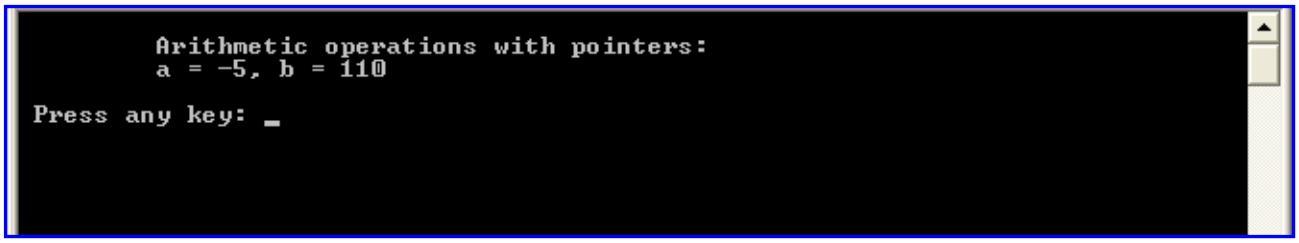
При выполнении примера следует иметь в виду, что операции **&** и ***** имеют более высокий приоритет, чем обычные арифметические операции.

Программный код решения задачи следующий:

```
#include <stdio.h>
#include <conio.h>
#include <stddef.h>
#include "stdafx.h"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
```

```
{
int x = 2, y = 7, a, b, *ptr, *ptr2;
ptr = &a;
ptr2 = &b;
*ptr = x - y;
*ptr2 = y - x - *ptr + 100;
printf("\n\t Arithmetic operations with pointers:\n");
printf("\t a = %d, b = %d\n", a, b);
printf("\n Press any key: ");
getch();
return 0;
}
```

Результат выполнения программы следующий:



Следует обратить внимание на то, что переменные **a** и **b** сначала не были определены, а в результате приобрели некоторые значения.

Задание 9.10. В программу для задания 2.9 внесите следующие изменения:

1. Примените типы данных **double** и **float**.
2. Напишите программу для выполнения операций вычитания, умножения и деления с применением указателей.

Задание 9.11. Разработать программу двухуровневой адресации для объектов целого типа.

Случай, когда указатель содержит адрес другого указателя, называется многоуровневой адресацией. При двухуровневой адресации первый указатель содержит адрес второго указателя, в котором находится адрес объекта с нужным значением. Объявление указателя на указатель делается с помощью **двух звездочек** перед именем переменной.

Программный код решения задачи имеет следующий вид:

```
#include <stdio.h>
#include <conio.h>
#include <stddef.h>
#include "stdafx.h"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int x, y = 8;
int *ptr, **ptr2;
x = 7;
ptr = &x;
ptr2 = &ptr;
**ptr2 = *ptr + 10;
printf("\n\t The value of x = %d. 1 st pointer is: %d. 2 nd pointer is: %d\n", x,      *ptr, **ptr2);
   ptr = &y;
```

```
    ptr2 = &ptr;
    **ptr2 = 88;
    printf("\n\t The value of y = %d\n", y);
printf("\n Press any key: ");
getch();
return 0;
}
```

Результат выполнения программы следующий:

Задание 9.12. Доработайте программу для задания 2.11:

1. Выведите на экран пользователя адреса указателей.
2. Организуйте цикл инкрементирования первого указателя, начиная с X до 10X, где X – номер компьютера, на котором выполняется лабораторная работа. Сделайте вывод значений переменной, на которую дает ссылку первый указатель, и значений второго указателя.
3. Напишите программу трехуровневой адресации при задании целых чисел, равных X и 10X, где X – номер компьютера, на котором выполняется лабораторная работа.

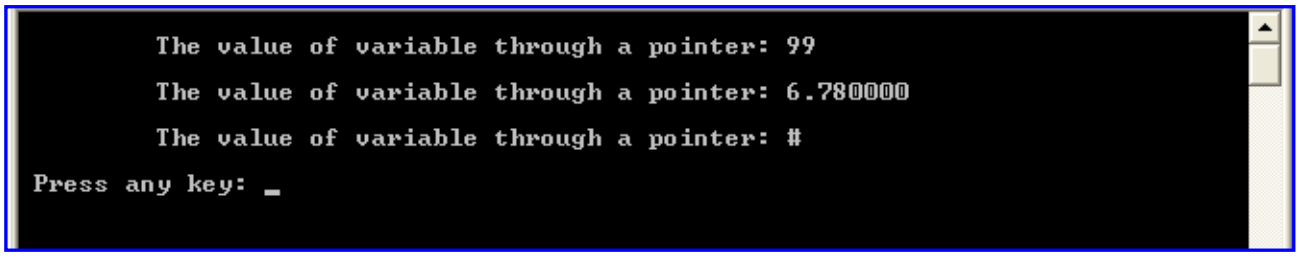
Задание 9.13. Разработать программу для определения и инициализации переменных разных типов и одного указателя типа **void.** Последовательно присваивая указателю адреса переменных, вывести значения переменных с помощью разыменования указателя.

Программный код для решения задачи следующий:

```
#include <stdio.h>
#include <conio.h>
#include <stddef.h>
#include "stdafx.h"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
int x = 99;
double y = 6.78;
char symbol = '#';
void *ptr;
ptr = &x;
printf("\n\t The value of variable through a pointer: %d\n", *(int *) ptr);
ptr = &y;
printf("\n\t The value of variable through a pointer: %lf\n", *(double *) ptr);
ptr = &symbol;
printf("\n\t The value of variable through a pointer: %c\n", *(char *) ptr);
printf("\n Press any key: ");
getch();
return 0;
```

}

Результат выполнения программы показан на рис. 9.5.



Особенностью использования указателя типа **void** является то, что при его разыменовании необходимо осуществлять преобразования типов. Прежде чем выполнить разыменование указателя, его приводят к указателю соответствующего типа.

Задание 9.14. Разработайте программный код на базе предыдущего, где:

1. Добавьте переменные типа **float**, **unsigned**, **long** и обеспечьте ввод их значений с клавиатуры. Выведите адреса и значения переменных с помощью разыменования указателя.
2. Задайте порядок (нумерованную последовательность) инициализации переменных и создайте вывод значений указателя на основе переключателя **switch**. Номер инициализируемой переменной задайте с клавиатуры.
3. Введите операцию двухуровневой адресации с применением указателя типа **void**. Выведите значения двух указателей с помощью их разыменования.

Задание 9.15. Разработайте программу для реализации следующего условия:

1. Определить и инициализировать переменную типа **double**.

2. Определить указатели типа **char ***, **int ***, **double ***, **void *** и инициализировать их адресом переменной.

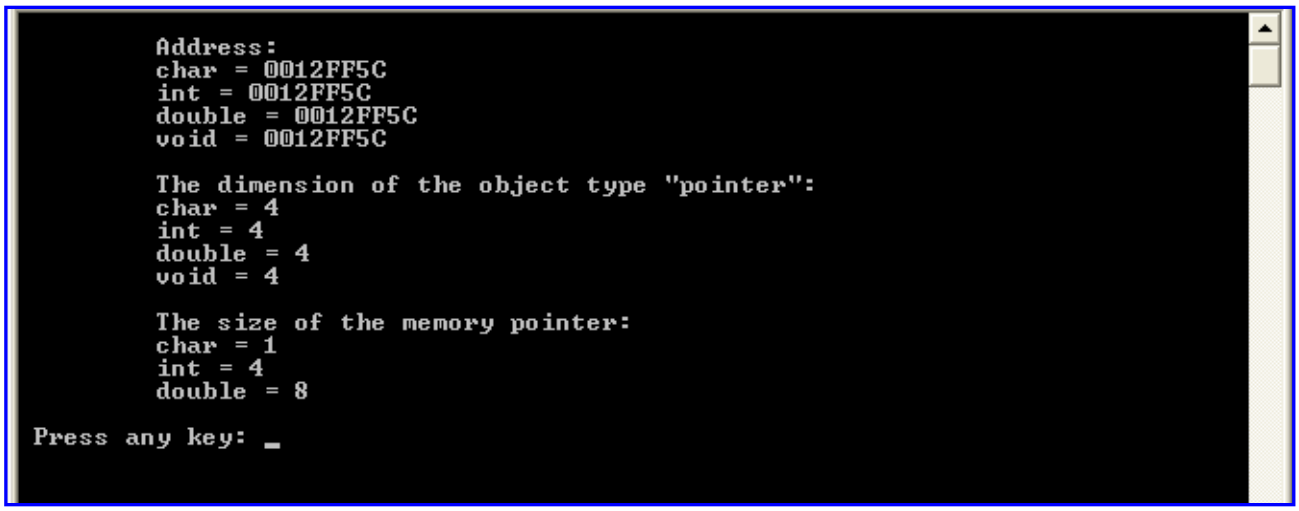
3. Вывести на экран пользователя значения указателей, их размеры и длины участков памяти, которые связаны с выражениями, разыменовывающими указатели.

Программный код для решения задачи следующий:

```
#include <stdio.h>
#include <conio.h>
#include <stddef.h>
#include "stdafx.h"
using namespace std;
int _tmain(int argc, _TCHAR* argv[])
{
double d = 6.78, *dp;
char *cp;
int *ip;
void *vp;
   //Адресация с приведением типов
cp = (char *)&d;
ip = (int *)&d;
dp = (double *)&d;
vp = &d;
printf("\n\t Address:\n\t char = %p\n\t int = %p\n\t double = %p\n\t void = %p\n", cp, ip, dp,
vp);
   //Размеры указателей и памяти разыменованных указателей:
```

```
printf("\n\t The dimension of the object type \"pointer\":\n\t char = %d\n\t int = %d\n\t double = %d\n\t void = %d\n",
sizeof(cp), sizeof(ip), sizeof(dp), sizeof(vp));
printf("\n\t The size of the memory pointer:\n\t char = %d\n\t int = %d\n\t double = %d\n",sizeof(*cp),sizeof(*ip),sizeof(*dp));
printf("\n Press any key: ");
getch();
return 0;
}
```

Результат выполнения программы следующий:



Как видно из полученного результата, размеры участков памяти, выделенных указателям разных типов, одинаковы.

Задание 9.16. Разработайте программу на базе предыдущей:

1. В программу добавьте вывод размера памяти для разыменованного указателя типа **void**.
2. Выведите значения указателей заданных типов. Определите указатель с правильным доступом к значению переменной **d = 6.78**.
3. Объявление указателей и взятие адреса сделайте в одной строчке для соответствующего типа.
4. В программу добавьте строки по вводу целого, вещественного типов данных, а также одиночного символа. Затем с помощью указателей выведите на консоль значения введенных данных.

Контрольные вопросы

1. Каково общее назначение указателей в языке **C++**?
2. Какие арифметические операции допускаются для указателей?
3. Какие унарные операторы используются с указателями? Как они называются?
4. Для каких типов данных может быть использован указатель?
5. Как числовые значения указателей изменяются при их инкрементировании в зависимости от типов данных.
6. С помощью какого формата осуществляется вывод на консоль адресов переменных заданного типа?
7. Что такое многоуровневая адресация? Как она организуется в языке **C++**?
8. Как осуществляется инициализация указателей на вещественные типы данных?
9. Как осуществляется инициализация указателей на символьный тип данных?
10. Какой смысл имеет значение указателя **NULL**?

11. Что произойдет, если применить к указателю со значением **NULL** операцию разыменования?
12. Как следует определять и инициализировать указатель на константу?
13. Как следует определять и инициализировать константный указатель?
14. В чем отличие константного указателя от указателя на константу?
15. Что будет выведено на экран после выполнения программы

int A=300;
cout<<&A;?

1. Значение **A**, то есть 300;
2. Адреса ячеек, в которых записано значение **A;**
3. Сообщение об ошибке.

16. Укажите правильное объявление указателя в С++.

1. ***double pA;**
2. **double *pA;**
3. **double pA*;**

17. Возможна ли следующая инициализация указателя:

char A=”yes”; char *pA=&A;?

1. Возможна;
2. Невозможна;
3. Такой конструкции в С++ нет.

18. Что означает оператор **double *pA=&A;**?

1. Инициализация переменной **A;**
2. Объявление указателя **pA**;
3. Инициализация указателя **pA** и присваивание ему адреса переменной **A.**

19. Что такое указатель в языке С++?

1. Адрес некоторой переменной или функции;
2. Переменная, содержащая адрес некоторой переменной;
3. Стандартная функция.

20. Как объявляется указатель с именем **pA** в языке С++?

1. **int** ***pA**;
2. **int** «**pA**;
3. **int &A**.

21. Как обозначается операция взятия адреса переменной **A** в языке С++?

1. ***pA**;
2. «**pA**;
3. **&A**.

22. Как обозначается операция разыменования указателя **pA** в языке С++?

1. ***pA**;
2. «**pA**;
3. **&pA**.

23. 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.

24. 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 **X;**
3. Будет выдано сообщение об ошибке с указанием на последнюю

строку.

25. Можно ли в языке С++ выполнять арифметические операции над указателями?

1. Можно;
2. Нельзя;
3. Только операцию присваивания.

26. 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 С++ использовать нельзя.

27. Укажите на возможность такого объявления указателя **int **ppA;**

1. Возможно;
2. Невозможно;
3. Все зависит от содержания программы.

28. 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