Лабораторна робота №4. Покажчики

Мета і задачі:

Навчитися створювати та відлагоджувати програми, в яких використовуються покажчики на мові програмування С.

Теоретичні відомості і методичні вказівки

Покажчик - це змінна, значенням якої є адреса іншої змінної. Так як покажчик може посилатися на змінні різних типів, з покажчиком у мові Сі зв'язується тип того об'єкта, на який він посилається. Для опису покажчиків використовується операція непрямої адресації *. Наприклад, покажчик цілого типу *uk* описується так:

```
int * uk;
```

Унарна операція &, застосована до деякої змінної, показує, що нам потрібен адрес цієї змінної, а не її поточне значення. Якщо змінна *uk* оголошена як покажчик, то оператор присвоювання *uk = & x* означає: "взяти адресу змінної *x* і привласнити його значення змінній-вказівником *uk*".

Унарна операція * застосована до покажчика, забезпечує доступ до вмісту комірки пам'яті, на яку посилається покажчик. Наприклад, * *uk* можна описати словами як "те, що міститься за адресою, на який вказує *uk*". Покажчики можуть використовуватися в виразах. Якщо. наприклад, змінна *uk* вказує на ціле *x*, то * *uk* може у всіх випадках використовуватися замість *x*; так, * *uk +1* збільшує *x* на одиницю, а * *uk = 0* рівносильне *x = 0*. Два оператора присвоювання *uk = &x; y =* *uk;* виконує те ж саме, що і один оператор у = х. Користь від застосування покажчиків в таких ситуаціях, м'яко кажучи, невелика.

Найбільш повно їх переваги при обробці масивів і, зокрема, символьних рядків. Покажчики та масиви тісно пов'язані один з одним. Перш ніж розглянути цей зв'язок детально, зазначимо, що якщо *uk* - деякий покажчик, то *uk* ++ збільшує його значення і він тепер вказує на наступний, сусідній об'єкт. Значення *uk* використовується у виразі, а потім збільшується. Аналогічно визначаються операції *uk*--, ++*uk*, --*uk*. У загальному випадку покажчик *uk* можна додавати до цілого числа *i*. Оператор *uk* += *i* пересуває посилання на *i* елементів щодо поточного значення. Ці конструкції підпорядковуються правилам адресної арифметики.

А тепер повернемося до масивів. Нехай є опис *int a[5]*. Воно визначає масив розміром 5 елементів, тобто п'ять послідовних розташованих комірок пам'яті *a[0]*, *a[1]*, *a[2]*, *a[3]*, *a[4]*. Адреса і-го елемента масиву дорівнює сумі адреси початкового елемента масиву і зміщення цього елемента на *i* одиниць від початку масиву. Це досягається індексуванням: *a[i]* - і-й елемент масиву. Але доступ до будь-якого елементу масиву може бути виконаний і за допомогою покажчиків, причому, більш ефективно. Якщо *uk* - покажчик на ціле, описаний як *int * uk*, то *uk* після виконання операції *uk = & a[0]* містить адресу *a [0]*, а *uk + i* вказує на і-й елемент масиву. Таким чином, *uk + i* є адресою *a[i]* комірки масиву, а * *(uk + i)* - вмістом і-го елемента (операції * і & є більш пріоритетними, ніж арифметичні операції). Оскільки ім'я масиву в програмі ототожнюється з адресою його першого елемента, то вираз *uk = & a [0]* еквівалентно такому: *uk = a*. Тому значення *a[i]* можна записати як * *(a + i)*. Застосувавши до цих двох елементів операцію взяття адреси, отримаємо, що *& a[i]* і *a + i* ідентичні.

Порядок виконання і звітування

1. Створити програму на мові С згідно варіанту використавши середовище програмування Dev-C++ 4.0:
    - для рішення задачі використати динамічний масив;
    - масив задати з клавіатури;
    - для рішення задачі написати функцію, яка буде оперувати елементами масиву. Параметри в функцію передавати через покажчик, крім окремо оговорених в варіантах завдань випадків.
2. Відкомпілювати та відлагодити програму.
3. Розробити набір тестів і перевірити роботу програми на них.
4. Відповісти на контрольні запитання.
5. Зробити висновки.
6. Звіт по лабораторній роботі має складатися з титульної сторінки, лістингів програм, висновків по роботі.

Варіанти завдань

Варіант 1.

Написати програму, яка перетворює масив таким чином, щоб спочатку розташовувалися нульові елементи, а потім всі інші.

Варіант 2.

Написати програму, яка видаляє всі елементи масиву, модуль яких не перевищує 1; елементи, які звільнилися в кінці масиву заповнюються нулями.

Варіант 3.

Написати програму, яка перетворює масив таким чином, щоб спочатку розташовувалися елементи, які відрізняються від максимального не більш ніж на 20%, а потім усі інші.

Варіант 4.

Написати програму, яка перетворює масив таким чином, щоб спочатку розташовувалися додатні числа, а потім від’ємні.

Варіант 5.

Написати програму, яка буде упорядковувати елементи масиву за зростанням модулів елементів.

Варіант 6.

Написати програму, яка буде змінювати порядок елементів в масиві на зворотній.

Варіант 7.

Написати програму, яка буде перетворювати масив таким чином, щоб нульові елементи розташовувалися після всіх інших.

Варіант 8.

Написати програму, яка видаляє всі елементи, модуль яких знаходиться в інтервалі [a,b]; елементи які вивільнилися в кінці масиву заповнити нулями.

Варіант 9.

Написати програму, яка перетворює масив таким чином, щоб в першій його половині знаходилися елементи, які знаходилися в непарних позиціях, а в другій половині – елементи які знаходилися в парних позиціях.

Варіант 10.

Написати програму, яка перетворює масив таким чином, щоб спочатку розташовувалися всі елементи, модуль яких не перевищує 1, а потім всі інші.

Підсумок

Після виконання лабораторної роботи студент повинен вміти створювати програми, в яких для обчислювальних процесів використовуються покажчики на мові програмування С.

Контрольні питання

1. Що таке покажчик ?

2. Назвіть три основні значення покажчика ?
3. Що повинно бути операндом операції ?
4. Що таке операція * ?
5. Назвіть чотири способи передачі покажчика в функцію ?
6. Які операції можуть виконуватися разом з покажчиками ?
7. Для чого призначений покажчик типу **void *** ?
8. Що таке ім'я масиву ?

Контрольні вправи

1) Що надрукує наступна програма?

```
# include <stdio.h>
int    Array [ ] = { 0, 4, 5, 2, 3 };
int main  ( void)
{
     int    Index, *Pointer;
     for ( Index = 0; Index <= 4; Index +=2 )
            printf ( " %3d", * (Array+Index--) );
     printf ( "\n" );
     Pointer = Array + 1;
     for ( Index = 0; Index <= 2; )
            printf ( " %3d", Pointer [ ++Index ] );
     printf ( "\n" );
     return 0;
}
```

2) Що надрукує наступна програма?

```
# include <stdio.h>
int    Array [ ] = { 1, 2, -7, 4, 3  };
int main  ( void)
{
     int    Index, *Pointer;
     for ( Index = 0; Index <= 4; Index +=2 )
           printf ( " %3d", Array[ Index] );
     printf ( "\n" );
     Pointer = Array + 1;
     for ( Index = 0; Index <= 2;  ++Index )
```

```
            printf ( " %3d", Pointer [ ++Index ] );
      printf ( "\n" );
      return 0;
}
```

3) Що надрукує наступна програма?

```
# include <stdio.h>
int    Array [ ] = { 1, 4, 7, 2, 3  };
int main  ( void)
{
      int    Index, *Pointer;
      for ( Index = 1; Index <= 4; Index +=1 )
            printf ( " %3d", Array[ ++Index] );
      printf ( "\n" );
      Pointer = Array ;
      for ( Index = 0; Index <= 2;  ++Index )
             printf ( " %3d", Pointer [ Index++ ] );
      printf ( "\n" );
      return 0;
}
```

4) Що надрукує наступна програма?

```
# include <stdio.h>
int    Array [ ] = { 1, 4, 5, 12, 3};
int main  ( void)
{
      int    Index, *Pointer;
      for ( Index = 1; Index <= 4; Index +=1 )
             printf ( " %3d", * (Array+Index++) );
      printf ( "\n" );
      Pointer = Array + 1;
      for ( Index = 0; Index <= 2;  ++Index )
             printf ( " %3d", Pointer [ ++Index ] );
       printf ( "\n" );
       return 0;
}
```

5) Що надрукує наступна програма?

```
# include <stdio.h>
int    Array [ ] = { 0, 4, 5, 2, 3  };
int main  (void)
{
     int    Index, *Pointer;
     for ( Index = 0; Index <= 4; Index +=2 )
           printf ( " %3d", * (Array+Index++) );
     printf ( "\n" );
     Pointer = Array + 1;
     for ( Index = 0; Index <= 3;  Index++ )
           printf ( " %3d", Pointer [ ++Index ] );
      printf ( "\n" );
      return 0;
}
```

Джерела інформації

1. http://void.net.ua/The_C_Programming_Language.html.
2. http://publications.gbdirect.co.uk/c_book/.
3. http://www.scribd.com/doc/16306895/Draft-ANSI-C-Rationale
4. http://www.cplusplus.com/doc/tutorial/