НЯКОИ ПРИЛОЖЕНИЯ НА МАСИВИ ОТ УКАЗАТЕЛИ

Елка Иванова

Изучаването на променливи-указатели, които за краткост ще бъдат наричани указатели, е трудна дейност поради тяхната специфика. За разлика от обикновените променливи те съдържат адреси и служат най-често за индиректно обръщение към обекти. Умението за боравене с тях е задължително условие, когато става дума за структури от данни, управление на паметта и обектно-ориентирано програмиране. Езикът за програмиране C/C++ може би е езикът, който в най-голяма степен използва гъвкавостта на този тип променливи. Изучаването им започва в курсове по програмиране и структури от данни и включва обсъждане на следните аспекти: деклариране, инициализиране, индиректно обръщение, аритметика с указатели, предаване на параметри по адрес, указатели към тип void, указатели-константи и указатели към константи, указатели и масиви, указатели и функции. Приложението им продължава в курсовете по обектно-ориентирано програмиране и е елемент на всеки реален програмен проект, което оправдава важността на темата указатели. Тук ще бъдат разгледани няколко примера за използване на масиви от указатели при изучаване на езика за програмиране C/C++. Поводът за това е, че често в учебници по програмиране или за даден език за програмиране не се споменава изобщо или пък само бегло възможното приложение на такава структура било то поради ограничен обем или други причини и липсват задачи за използването им, като например в [4] и [5]. Така възможното им подходящо приложение остава неразгледано. Целта на този доклад е да направи, макар и непълен, списък на ситуации, в които масивите от указатели биха били най-подходящата структура. Посочените примери могат да бъдат използвани от всички преподаватели по програмиране от ВУЗ и от учителите по информатика.

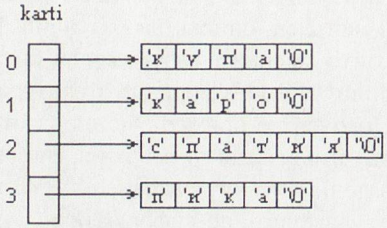
Най-разпространената употреба на масиви от указатели е за формиране на масив от низове в C. Всеки елемент на такъв масив е низ, но както знаем, в C низовете се представят чрез указател към първия им символ.Така че масив от низове е всъщност еквивалентен на масив от указатели към техните начални символи. Ето един пример за масив от низове - наименованията на четирите цвята карти за игра:

```
char   *karti[4] =
{“купа”, “каро”, “спатия”, ”пика”};
```

При липса на опит за използване на указатели е възможно допускане на грешка при декларирането, а именно поставяне на скоби по следния начин:

```
char  (*karti)[4]
```

Понеже [ ] има по-висок приоритет от *, то това ще бъде декларация на указател към масив от низове). Масивът karti има елементи - указатели към низове. Всеки от тези низове завършва със символа за край на низ '\0'. Въпреки че изглежда, че в масива са записани целите низове, всъщност той съдържа само указателите към техните начални символи:



Въпреки че масивът karti е с фиксиран размер, той позволява достъп до символни низове с различна дължина (едно от мощните предимства на C).

Масивът karti може да бъде поместен в двумерен масив, на който всеки ред представя един цвят, а всяка колона - една от буквите от наименованието на съответния цвят. Недостатъкът на това представяне е, че броят на колоните за всеки ред е фиксиран и е равен на броя на символите на най-дългото име на цвят + 1 (за нулевия байт).

Между другото инициализирането на масив от указатели е идеално приложение на вътрешен статичен масив:

```
function( )
{ static  char  *days = {"понеделник", "вторник", "сряда", ...};  }
```

Задача: Да се сортират редове от текст с различна дължина (Забележка: Текстът не може да бъде местен или сравняван в една операция).

За решаването на тази задача ефективни се оказват масиви от указатели (това е едно от най-полезните им приложения). Алгоритъмът е следният:

1. Съхраняват се редовете в един масив от тип char. Така всеки ред ще завършва с нулев символ.

2. Съхраняват се указателите в отделен масив, като всеки указател сочи първия символ на всеки нов ред.

3. Сравняват се два реда, като се използва стандартната библиотечна функция strcmp ( ).

4. Ако двата реда не са подредени, разменят се местата на съответните им указатели в масива с указатели (не се разменят местата на редовете с текста).

Този алгоритъм елиминира сложното манипулиране с памет и неефективното разместване на цели редове.

Популярен пример за използване на масив от указатели (от низове) е този за размесване по случаен начин на тесте (колода) от 52 карти ([1]). Всеки цвят карти има 13 разновидности (асо, двойка, тройка, ... , дама, поп) и затова може да се използва двумерен масив с 4 реда и 13 колони за представяне на всичките карти, масива karti, който вече бе деклариран и масив vid [13] за различните видове карти.

Когато се разглеждат структури от данни като стек, линеен списък, опашка и т. н., обикновено се използва реализацията им, при която данните на елементите са записани във възлите. При определени ситуации дадена линейна структура може да се представи като масив от указатели, които сочат реалните данни.

Когато е необходимо да се създаде многомерен масив в динамичната памет, най-добрият начин е да се създаде масив от указатели и след това всеки указател да се инициализира с адреса на динамично резервиран "ред". Ето пример за динамично създаден двумерен масив:

```
#include <stdlib.h>

int **array1 =
(int **)malloc(nrows * sizeof(int *));
for(i = 0; i < nrows; i++)
array1[i]=(int*)malloc(ncolumns*
sizeof (int));
```

(За краткост е пропусната проверката за стойностите, които връща функцията malloc.) Продължаваме да реалокираме отделните редове на масива, като използваме аритметика с указатели, за да се движим последователно по масива:

```
int **array2 =
(int **)malloc(nrows * sizeof(int *));
array2[0] = (int *)malloc(nrows *
ncolumns * sizeof(int));
for(i = 1; i < nrows; i++)
array2[i] = array2[0] + i * ncolumns;
```

Елементите на динамичен масив могат да бъдат достигнати по подобие на нормален масив:

```
arrayx[i][j] ( 0 <= i < nrows ,
0 <= j < ncolumns).
```

Масивите от указатели се използват и при създаването на програми с меню, което позволява избирането на една от няколко възможни опции. Всяка опция се обслужва от различна функция. Указателите към всяка функция се съхраняват в масив. Изборът на потребителя определя кой от елементите на масива ще бъде използван, а този елемент (той е указател към съответната функция) я стартира. Ето един пример за меню с три функции - func1, func2, func3. Всяка от тях има един аргумент от тип int и не връща стойност. Указателите към тези функции се съхраняват в масив f, деклариран по следния начин:

```
void (*f[3])(int) =
{func1, func2, func3};
```

```
/* Програма за демонстриране
на масив от указатели към функции
*/
#include <stdio.h>

void func1(int a)
{  printf("Извикана е
func1\n\n", a);  }

void func2(int b)
{  printf("Извикана е
func2\n\n", b); }

void func 3(int c)
{  printf("Извикана е
func3\n\n", c); }

main()
{  void (*f[3])(int) =
{func1, func2, func3};
   int choice;

   printf("Изберете 0, 1 или 2;
Въведете 3 за край: ");
   scanf("%d", &choice);
   while (choice >=
   0 && choice < 3) {
     (*f[choice])(choice);
     printf("Изберете 0, 1 или
2; Въведете 3 за край:  ");
      scanf("%d", &choice);
   }
printf("Въведено е 3 за край\n");
return 0;
  }
```

Друго приложение на масив от указатели е използването на функцията main(), но с аргументи. Ако се очаква вход от командния ред, функцията използва специалните аргументи argc и argv[ ] за достъп до въведеното и след това се обръща към други функции. Известно е, че тази функция има следните две форми:

```
int main()
{/* ... */}

или

int main
( int argc, char* argv[ ] )
{/* ... */}
```

Тук **argc** е броят на параметрите, предадени на програмата. Ако **argc** не е нула, параметрите се предават като низове, завършващи с нулев байт, съответно елементи на масива от указатели **argv[]**, чиито брой елементи е **argc**; **argv[0]** съдържа името на програмата, като трябва **argv[argc]** = 0. Например нека целта е да се напише конзолно приложение или такова за DOS, наречено Sort.EXE, което приема два аргумента от конзолния ред - име на входен файл и име на изходен файл:

```
Sort  data.txt  result.txt
```

Аргументът argc съдържа общия брой елементи на командния ред, в дадения случай 3. Масивът от указатели argv[ ] ще съдържа три указателя, които ще сочат към низовете "sort", "data.txt" и "result.txt".

Разгледаните примери далеч не претендират за пълнота и изчерпателност, но се предполага, че ще бъдат полезни за демонстрация или самостоятелно усвояване на принципите и техниките за използване на указатели.

ЛИТЕРАТУРА

1. *Deitel & Deitel.* C How to program, 1994, PRENTICE HALL, New Jersey
2. *Овърленд, Б.* C++ на разбираем език, 1997, IDJ Books Worldwide, Inc.
3. *Смит, Т. М.* Програмиране с Pascal, принципи и методи. С., Техника, 1996 г.
4. *Георгиева, Ю., И. Йорданов, М. Коранова, С. Малешков.* Ръководство по програмиране I (C). С., Сиела, 1998г.
5. *Тодорова, М.* Програмиране на Паскал. С., 1995 г.

SOME PRACTICAL APPLICATIONS OF ARRAYS OF POINTERS

ELKA IVANOVA

Summary

This paper considers some examples for practical applications of arrays of pointers in teaching pointers during a programming language course (in this case C++) . Arrays of Pointers are a data representation that will cope efficiently and conveniently with variable length text lines. They are useful for dynamically allocating of multidimensional arrays, for demonstrating a common use of function pointers in so-called menu driven systems, for passing command-line arguments when a C++ program is run. Mastering pointers comes naturally from following a few basic principles and techniques and it is very useful for good programming. All considered examples could be a good exercise.