Лекція №17

Тема лекції: Вказівники на функції

План лекції

1. Оголошення вказівника на функцію
2. Звертання через вказівник на функцію
3. Вказівник на функцію як параметр функції
4. Деякі приклади з використанням вказівників на функцію
5. Функції з невизначеною кількістю параметрів

Зміст лекції

1. Оголошення вказівника на функцію

Ідентифікатор функції є константним вказівником на перший байт виконуваного коду функції і визначає **точку входу** у функцію. Коли процесор передає управління на функцію, то фактично виконання програми буде продовжено з даної адреси.

Так само як і для масиву, можна визначити вказівник на функцію, який уже не буде константним і з яким можна буде працювати. Синтаксис оголошення такого вказівника відрізняється від оголошення вказівників на змінні і виглядає так:

```
тип(*ідент.вказ.)([список типів параметрів]);
```

Вказівник на функцію вводиться окремо від оголошення та визначення будь-якої функції і визначає, до якої саме групи функцій може бути застосований, а саме: тільки до функцій із заданою схемою формальних параметрів і типом результату. Наприклад оголошення

```
double (*pf)(int, double*);
```

визначає, що змінна **pf** може бути використана як вказівник на будь-яку функцію, що повертає **double** і має два параметри – **int** та **double***. Порядок розташування параметрів має значення.

Звертання через вказівник на функцію

Вказівнику на функцію може бути присвоєний інший вказівник на функцію зі схожим прототипом. Наприклад:

```
void (*pf)(void);  // вказівник на функцію без  параметрів
void (*qf)(void);
qf=pf;
```

Вказівнику на функцію можна присвоїти значення відповідної функції:

```
ідент.вказівника=ідент.функції;   або
ідент.вказівника=&ідент.функції;
```

Наприклад:

```
pf=sum;
pf=&sum;
```

Вказівник на функцію – це змінна, що містить адресу деякої функції, тому може бути використаний для виклику цієї функції. Синтаксис виклику функції за допомогою розадресації вказівника (перший варіант):

```
(*ідент.вказівника)(список фактичних параметрів);
```

Такий варіант виклику явно вказує на застосування саме вказівника на функцію, а не просто функцію.

У сучасних компіляторах допустимий також інший різновид виклику, схожий на виклик звичайної функції, (другий варіант):

```
ідент.вказівника(список фактичних параметрів);
```

Але краще дотримуватись першого варіанту виклику, щоб код програми був зрозуміліший для людини.

Наприклад:

```
void print(int n){ /* реалізація          */   };
. . .
void (*qf)(int);      // оголошення вказівника в main()
qf=print;         // тотожно qf=&print;
(*qf)(10);              // перший варіант з розадресованим вказівником
qf(10);                 // другий варіант виклику
```

Арифметичні операції над вказівниками на функції заборонені.

Вказівник на функцію як параметр функції

Вказівники на функції часто використовують як параметр іншої функції. Замість того, щоб розробляти декілька різних великих функцій, відміни яких – це незначні деталі, які можуть бути оформлені як окремі декілька функцій,

прототипи яких відрізнятимуться лише ідентифікатором, можна цю відмінність передати як параметр функції.

Наприклад, потрібно написати функцію визначення максимального або мінімального елемента масиву, у яку як параметри передаються: сам масив, його розмірність і вказівник на функцію для порівняння.

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define n 10
double less(double a, double b){
     if(a<b) return a;return b;
}
double grt(double a, double b){
     if(a>b) return a;return b;
}
void gen(int n, double a[]){
int i=0;
     for(;i<n;i++)a[i]=0.01*(rand()%100);
}
double extrem(int n, double *a, double(*pf)(double, double)){
double max=a[0];
int i=1;
    for(; i<n; i++) max=(*pf)(max,a[i]);
    return max;
}
void out(int n, double*p){
    while(n-1){
          printf("%5.2f ",*p);
          p++;n--;
    };
    printf("\n");
};
int main(){
    srand(time(0));
    double a[10], m;
    gen(n,a);
    out(n,a);
    m=extrem(n, a, less);
    printf("min=%lf\n",m);
    m=extrem(n, a, grt);
    printf("max=%lf\n",m);
return 0;
}
```

У наведеному прикладі функція **extrem** викликана двічі – один раз з підстановкою функції **less** у якості фактичного параметра, другий раз – функції **grt.**

Деякі приклади з використанням вказівників на функцію

Деякі обчислювальні алгоритми зручно реалізувати як окремі функції. Це алгоритми, відомі зі шкільного курсу математики. Вони цікаві тим, що можуть бути розглянуті як приклад використання вказівників на функції в якості параметрів іншої функції.

Наприклад, пошук кореня рівняння, яке складно розв'язати математично, методом ділення відрізка навпіл. Це так званий *метод дихотомії.*

Постановка задачі. Відомий відрізок, де може знаходитись тільки один з коренів заданого рівняння **[a,b].** Знайти цей корінь з точністю **ε**. Рівняння задане у вигляді **f(x)=0.**

Ідея методу. Відрізок ділимо навпіл і перевіряємо, чи можна вважати отриману точку **c** коренем рівняння з точністю **ε**, тобто що **|f(c)|<ε**. Якщо так, потрібний корінь з наданою точністю знайдено, якщо ні – обираємо новий відрізок, де знаходиться шуканий корінь – **[a, c]** або **[c, b],** а саме той з них, на граничних точках якого функція **f** матиме різні знаки і переходимо до нової ітерації.

Реалізація такої функції може мати вигляд:

```
double root(double a, double b, double eps,
                                    double(*f)(double)){
  double c, ff;
  char ch;
  a=a<b?a:b;
         do{
              c=(b-a)/2+a;
              ff=(*f)(c);
              if((*f)(a)*ff>0)a=c; else b=c;
         }while(fabs(ff)>=eps);
 return c;
}
```

Фактичним параметром у викликах такої функції може бути будь-яка стандартна або користувацька функція з відповідним прототипом.

Функції зі змінною кількістю параметрів

Список формальних параметрів може закінчуватися трьома крапками (...), що означає, що число аргументів функції змінне: на початку списку – декілька обов'язкових параметрів, після яких довільна кількість необов'язкових.

Синтаксис оголошення наступний:

`тип ідент.функції(сп.обов'язкових параметрів,...)`

Виклик такої функції повинен містити принаймні стільки обов'язкових аргументів, скільки формальних параметрів задано перед останньою комою у списку параметрів. Під час виклику компілятор перевірятиме узгодження типів тільки для обов'язкових параметрів, оскільки для необов'язкових така інформація відсутня.

Прикладами таких функцій можуть бути функції **`printf()`** і **`scanf()`.**

За перевірку типів і організацію коректної роботи з параметрами такої функції повністю відповідає програміст, кількість параметрів передають до функції через один із обов'язкових аргументів.

Словник. Точка входу, вказівник на функцію, тип функції, функція зі змінною кількістю параметрів, обов'язкові і необов'язкові параметри.

Завдання на СРС:

1) Масиви вказівників на функцію[6, с.137]
2) Функції, що повертають вказівник на функцію[2,с.77, 3, с.138]

Контрольні запитання

1. Що таке точка входу функції?
2. Як оголосити вказівник на функцію? Які значення можна йому присвоїти?
3. Які правила оголошення функції зі змінною кількістю параметрів?