Ukazatele na funkce

1 Funkce

K pochopení této části je potřeba správné pochopení ukazatelů na proměnné a ukazatelů na ukazatele. Tato část se bude totiž věnovat ukazatelům na funkce. V jazyce C nejsou funkce proměnné, ale lze na ně ukazovat a tyto ukazatele pak můžeme přiřazovat, ukládat do polí i vracet jako výsledek funkce.

Při definici funkce, je tato funkce uložena někde v paměti, existuje tedy adresa, která ukazuje na začátek této funkce. Pomocí ukazatele na funkci lze funkci zavolat (a to i s příslušnými argumenty). Díky tomu je možné napsat funkci, která bude mít jako argument ukazatel na jinou funkci, kterou je pak možné spustit.

1.1 Motivace

Jako motivaci, k čemu tohoto můžeme využít, si vezmeme příklad třídění.

Úlohu třídění můžeme rozdělit do 3 částí:

- porovnávání - určení vzájemné pozice pro dva prvky
- výměna - obrácení pořadí dvou prvků
- třídící algoritmus, který provádí porovnání a výměny

Třídící algoritmus je na operacích porovnání a výměny nezávislý. Změnou porovnávací funkce (případně i funkce na výměnu) dostaneme algoritmus třídící podle jiných kritérií.

1.2 Deklarace

Při deklaraci proměnné typu ukazatel na funkci je nutné vždy specifikovat také počet argumentů, jejich datové typy a návratovou hodnotu.

```
typ (*nazev)(typ1, typ2, ..., typn);
```

Proměnná `nazev` je ukazatel na funkci s n parametry (typu po řade $typ1$, $typ2$, ..., $typn$) a návratovou hodnotou typ.

Závorky kolem jména funkce a hvězdičky jsou nutné. Pokud by se vynechali, byla by to deklarace funkce, která má jako návratový typ `typ *`.

Jako konkrétní příklad si ukážeme definici funkce `polynomA` a vytvoření ukazatele na tuto funkci.

```
double polynomA(double x)
{
    return 3*x*x + 2*x + 1;
}

double (*ptr_polynom)(double) = polynomA;
```

Identifikátor přiřazované funkce (v tomto případě `polynomA`) se musí uvádět bez závorek se seznamem parametrů. I přes to, že chceme pointeru přiřadit adresu funkce, tak se zde nepoužívá `&` pro získání adresy, je to proto, že kompilátor dokáže rozeznat, že chceme přiřadit ukazatel na funkci a ne funkci samotnou. Přiřadit proměnné přímo funkci totiž v C není možné.

Pro usnadnění práce s ukazateli na funkce můžeme využít `typedef` pro pojmenování datového typu ukazatele na funkci.

```
typedef double (*PTR_FUN)(double);

PTR_FUN ptr_polynom = polynomA;
```

1.3 Práce s ukazateli na funkce

Přiřazení adresy ukazateli:

```
ptr_polynom = polynomA;
```

Volání funkce pomocí ukazatele:

```
v = ptr_polynom(-1);

// pripadne
v = (*ptr_polynom)(-1);
```

Tyto možnosti se neliší. Je možné používat obě. První je ale jednodušší, proto jí budu používat. Výsledkem vyhodnocení tohoto výrazu bude double hodnota 2, stejně, jako bychom zavolali funkci `polynomA(-1)`, na kterou ukazatel ukazuje.

1.4 Příklad

`pr8-polynom.c`

```
#include <stdio.h>

double polynomA(double x)
{
    return 3*x*x + 2*x +1;
}

double polynomB(double x)
{
    return -3*x*x + 2*x +1;
}

// double (*ptr_polynom)(double) = polynomA;

typedef double (*PTR_POLYNOM)(double);

int main(int argc, char* argv[])
{
    PTR_POLYNOM ptr_polynom = polynomA;
    PTR_POLYNOM ptr_polynom2 = polynomB;

    printf("polynomA: %f \n", ptr_polynom(2));
    printf("polynomB: %f", (*ptr_polynom2)(2));
    return 0;
}
```

2 Ukazatel na funkci jako parametr funkce

Funkcím, které jako argument berou jiné funkce, říkáme **funkce vyšších řádů**. Výpočet těchto funkcí je možné za běhu programu modifikovat pomocí funkcí, které jim jsou při jejich volání předány jako parametr.

Deklarace:

```
double *map(double (*fce)(double), double *vstup, int pocet)
{
    ...
}
```

Tato funkce bere jako argument funkci, která má být aplikovaná na pole vstupních hodnot typu double, vstupní hodnoty a jejich počet.

Volání této funkce:

```
/* Funkce vracejici treti mocninu prvku x */
double na3(double x)
{
    return x*x*x;
}

pole_vysledku = map(na3, pole_vstupni, velikost_pole);
```

Celý kód definice a použití funkce map (`pr8-ukazatel-parametr.c`):

```
#include <stdio.h>

double *map(double (*fce)(double), double *vstup, int pocet){
    int i;
    double *vystup = malloc(pocet*sizeof(double));
    for(i=0;i<pocet; i++)
    {
        vystup[i]=fce(vstup[i]);
    }
    return vystup;
}
```

```
double na3(double x)
{
    return x*x*x;
}


int main(int argc, char* argv[])
{
    double pole[5] = {0.1, 0.2, 2, 3, 0.5};
    double *pole_vysledku = map(na3, pole, 5);

    for(int i = 0; i<5; i++)
    {
       printf("%f \n", pole_vysledku[i]);
    }

    free(pole_vysledku);
    return 0;
}
```

3 Pole ukazatelů na funkce

Pokud máme funkce stejných typů (se stejnou návratovou hodnotou), které berou stejné vstupní argumenty, můžeme tyto funkce uchovávat v poli.

```
/* Deklarace */
double (*pole_fci[5])(double);

/* Deklarace + inicializace */
double(*pole_fci[5])(double) = {na0, na1, na2, na3, na4};

/* volani */
vysledek = pole_fci[1](-1);
```

Pole se definuje a inicializuje stejně jako pole jiného typu. K jednotlivým prvkům pole se přistupuje přes hranaté závorky (stejně jako u jiných polí).

4 Cvičení

1. Doprogramujte příklad z předchozí kapitoly (naprogramujte funkce `na0`, `na1`, `na2`, `na3` a `na4`) a vyzkoušejte si práci s tímto polem.
2. Využijte předchozí příklad a funkci map a vypište vždy prvních 10 nultých mocnin, prvních mocnin, ..., čtvrtých mocnin.
3. Naprogramujte obecnou funkci pro třídění s tím, že jí je možné předat jako parametry funkce funkci porovnání a funkci výměny.
4. Vytvořte pole aritmetických funkcí (sčítání, odečítání, násobení...), naprogramujte tyto funkce. Napište program, který se zeptá uživatele na 2 vstupní hodnoty a operaci, kterou má se vstupy udělat a vrátí výsledek této operace.
5. Napište funkci `double akumulator(double (*fce)(double, double), double cisla[], int pocet)` , která zpracuje pomocí předané funkce `fce` hodnoty z pole `cisla`, jehož velikost je dána parametrem `pocet`. Vytvořenou funkci otestujte ve funkci `main()`. Použitými akumulačními funkcemi mohou být například funkce pro součet nebo součin dvou reálných čísel, které je ovšem pro testování potřeba dodefinovat. Detaily najdete zde:

   `http://jazykc.inf.upol.cz/ukazatele-na-funkce/akumulator.htm`