pole a/versus pointer

- z uvedených vlastností ukazatelů a typů je výjimka – proměnná typu pole
- pole a ukazatel mají natolik podobný přístup, že pokud je to možné, pole se konvertuje na ukazatel a dále se s nimi manipuluje stejně
- základní vlastností je, že název pole se konvertuje na ukazatel na první prvek pole („ztrácí“ se tím pojem o délce (?), kontrola opuštění mezí není ani v jednom případě)
- další vlastností je možnost „oindexování“ ukazatele (k adrese se přičte hodnota indexregistru – dojde k posunu na jinou adresu).
- díky znalosti typu ukazatele a tím i velikosti daného typu je krokem posunu v poli (i adresy ukazatele) vzdálenost rovná rozměru daného typu -> posunujeme se po prvcích pole
- první prvek má offset nula, druhý jedna ...
- index může být i záporný
- ukazatel je možné měnit (posunout), „hodnota“ pole je konstanta

Přístup k prvkům pole

- pro přístup k prvkům pole slouží unární operátor [ ] – parametrem je celočíselná hodnota udávající prvek od počátku (vztažná adresa)
- další možností je využití tzv. ukazatelové aritmetiky. Přičte-li (odečte-li) se k ukazateli celé číslo, výsledkem je adresa tolikátého prvku pole.

```
int Pole[10]={0,1,2,3,4,5,6,7,8,9}; // pole o deseti prvcích, indexy 0-9
int *pi = NULL; // pomocný ukazatel na int

pi = Pole; // díky konverzi ukazuje nyní pi na první (nultý) prvek Pole
Pole[2] = -2;
// přístup ke třetímu prvku pomocí operátoru [ ] – pracuje s hodnotou na dané pozici
*(Pole + 2) = -2; // ekvivalentní zápis – posun ukazatele o dvě „délky“ typu int
// a přístup k dané adrese. Závorky () jsou nutné kvůli větší prioritě * před +

*(pi + 2) = -2;
pi[2]=-2;
// přístup pomocí ukazatele je stejný. Nedochází ke konverzi z typu pole
```

S ukazateli je spojen pojem *ukazatelová aritmetika*

- přičteme-li (odečteme) k ukazateli celé číslo, „posune“ se adresa o daný počet prvků typu, na který ukazatel ukazuje (tj. N * sizeof(typ pole) bytů)
- přičtením (odečtením) jedničky se dostáváme na další (předchozí) prvek
- odečteme-li dva ukazatele (musí patřit ke stejnému poli/paměťovému celku) získáme počet prvků, které mezi nimi leží
- další matematické operace nejsou pro ukazatele definovány

```
int Pole[20]={0};
int *první, *aktualni,*poslední,i,pocet;// tři ukazatele na int a dva inty
int suma=0;
for (aktualni =první=Pole, i=0;i<20;i++,aktualni++) //posun ukazatele na další prvek
      suma += *aktualni;  // součet prvků v poli – nahraďte pro Pole a prvni
pocet = aktualni – první; //počet prvků mezi ukazateli.
// Prvky jsou typu, na který ukazují ukazatele (zde je to int)
```

přístup k poli přes ukazatel pro Pole a prvni

suma += Pole[i];

suma += *(první + i);

pole – předávání do funkce a z funkce – může být problém při kombinacích [] a * například pole[] na zásobníku – krátký relativní od zásobníku, ukazatel – dlouhý od počátku, nemusí dojít ke správné konverzi – dlouhý ukazatel na krátký od zásobníku

Zjištění velikosti pole

- pro zjištění velikosti typu se používá sizeof
- pokud je parametrem sizeof pole, vrací velikost pole, pokud je parametrem ukazatel, vrací velikost ukazatele (tj. adresy)

Co se stane v následujícím případě? Předává se pole jako pole nebo pomocí ukazatele?:

```
int Test(int pole[ ]) // popřípadě (int pole[10]) či (int *pole)
{
  int i;
  pole[0] = 10; // dojde ke změně hodnoty i mimo funkci?
  i = sizeof(pole);  // i nabývá hodnoty  ???
  return i;
}
```

```
int  Pole[20]= {1},i;
i = sizeof(Pole); // i nabývá hodnoty ??? Pole[0] nabývá hodnoty ???
i = Test(Pole); // i nabývá hodnoty ??? Pole[0] nabývá hodnoty ???
```