Základy programovania 25ZS
Pointer a Polia

Prednášajúci: prof. Gabriel Juhás
Cvičiaci: Milan Mladoniczky

Pointer *

- Obsahuje adresu nejakej informácie daného typu
- Operátor '**&**' (operátor adresy) získa adresu premennej
- Operátor '*' (operátor dereferencie) získa hodnotu na danej adrese

typ *pointer = &premenná

```
int cislo = 42;

printf("adresa: %d",  &cislo);  -> 24650

printf("hodnota na adrese: %d",  *(&cislo));  -> 42
```

Pointer *

- Obsahuje adresu nejakej informácie daného typu
- Operátor '**&**' (operátor adresy) získa adresu premennej
- Operátor '*' (operátor dereferencie) získa hodnotu na danej adrese

typ *pointer = &premenná

```
int cislo = 42;

printf("adresa: %p",  &cislo); -> 24650

printf("hodnota na adrese: %d",  *(&cislo));  -> 42
```


digit
42
24650

- Pointer musí byť definovaný na nejakom dátovom type (int, char a pod.), ktorého adresu ukazuje.
- ! Pointer neobsahuje hodnotu daného dátového typu ale adresa uložená v pointri ukazuje na hodnotu definovaného dátové typu !
- Pointer premenná vždy obsahuje adresu, ktorá má veľkosť 64 bitov alebo 32 bitov v závislosti na operačný systém.
- Podľa konvencie je pointer vypísaný ako hexadecimálne číslo (pre jednoduchosť môžeme použiť aj desiatkovú sústavu).

Divoký Pointer (Wild pointer)

- Definovaná pointer premenná bez hodnoty.
- Pointer tak ukazuje na náhodnú adresu pamäte, ktorá nevieme či je rezervovaná nášmu programu.
- Veľmi nebezpečné používať divoké pointre!

```
char *adresaZnaku; -> divoký pointer

char znak = “a”;

adresaZnaku = &znak; -> tu už nie je divoký pointer
```

Null Pointer

- Zabránenie vzniku divokých pointrov.
- Pointer hned inicializujeme na hodnotu NULL (teda hodnota ničoho)
- Pointer tak nemá žiadnu adresu a je bezpečné s ním pracovať ďalej.

```
char *adresaZnaku = NULL
```

Void Pointer / Generic Pointer

- Pointer môže obsahovať adresu na premennú akéhokoľvek typu.
- Pred použitím pointra je však nutné jeho pretypovanie na požadovaný dátový typ.

```
void *pointer = NULL;

int cislo = 54;

pointer = &cislo;
printf("hodnota: %d", *pointer); -> Chyba kompilácie
printf("hodnota: %d", *(int *)pointer); -> 54
```

Visiaci Pointer (Dangling Pointer)

- Pointer ktorý bol voľakedy použitý na uloženie adresy,
ale už bol uvoľnený a jeho hodnota je nepredikovateľná.

```
int *ptr;

ptr = (int *)malloc(sizeof(int));  -> alokácia priestoru pre číslo

*ptr = 1;

printf("%d",*ptr); -> vypíše 1

free(ptr);   -> uvoľnenie pamäte

*ptr = 5;

printf("%d",*ptr); -> neviem čo vypíše, môže nemusí to byť 5
```

Aritmetika pointerov

- Priradenie hodnoty pointra je možné do iného pointra jedine rovnakého typu.
- Pričítanie a odčítanie čísla (int) k pointra posunie jeho hodnotu, t.j. adresu na ktorú ukazuje, o veľkosť dátového typu pointra.
- Vždy je možné porovnať a priradiť pointer k hodnote NULL

```
int cislo = 5;

int *ptr = &cislo; -> adresa je 100

int novaAdresa = ptr + 3;  -> nová adresa je 112

 /* Pretože výpočet = ptr + 3 * sizeof(int) */
```

Praktická ukážka –>

Polia

- Kolekcia / bloky pamäte za sebou idúcich hodnôt rovnakého dátového typu.
- Elementy pola je možné získať pomocou uvedenia ich indexu a operátorom '**[ ]**'.
- Premenná pole (t.j. adresa pola) je pointer na prvý element pola.
- Pointer na celé pole je získaný pomocou operátora **&**

```
int prime[5] = {2,3,5,7,11};

printf("&prime = %d",  &prime); -> 6422016

printf("prime = %d", prime);  -> 6422016

printf("&prime[0] = %d",  &prime[0]); ->
6422016
```

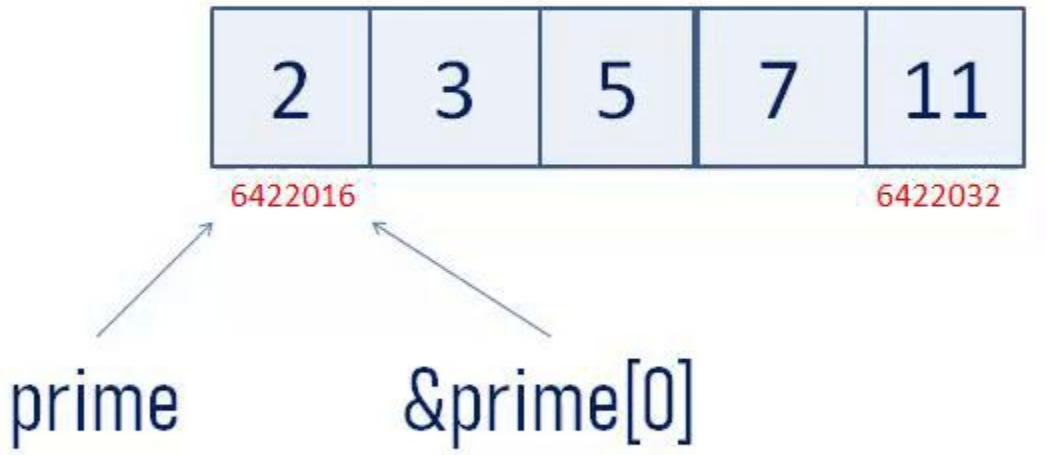
```
printf("&prime = %d",  &prime + 1);

printf("prime = %d",  prime + 1);

printf("&prime[0] = %d",  &prime[0] + 1);
```

```
printf("&prime = %d",  &prime + 1); -> 6422036

printf("prime = %d",  prime + 1);

printf("&prime[0] = %d",  &prime[0] + 1);
```

```
printf("&prime = %d",  &prime + 1); -> 6422036

printf("prime = %d",  prime + 1); -> 6422020

printf("&prime[0] = %d",  &prime[0] + 1);
```

```
printf("&prime = %d",  &prime + 1); -> 6422036

printf("prime = %d",  prime + 1); -> 6422020

printf("&prime[0] = %d",  &prime[0] + 1); -> 6422020
```

```
printf("&prime = %d",  &prime + 1); -> 6422036 (+ 20)

printf("prime = %d",  prime + 1); -> 6422020 (+ 4)

printf("&prime[0] = %d",  &prime[0] + 1); -> 6422020 (+4) == prime[1]
```


2
3
5
7
11
6422016
6422032
prime
&prime[0]

Prechádzanie pola cez for cyklus

```
int prime[5] = {2,3,5,7,11};

for( int i = 0; i < 5; i++)

{

  printf("index = %d, address = %d, value = %d\n", i,&prime[i], prime[i]);

}
```

Prechádzanie pola cez for cyklus

```
int prime[5] = {2,3,5,7,11};

for( int i = 0; i < 5; i++)

{

  printf("index = %d, address = %d, value = %d\n", i, prime + i,*(prime + i));

}
```

Reťazce (String)

- String / reťazec je pole znakov (t.j. pole typu char) termínované znakom `null(\0)`.
- Reťazec vieme teda zapísať ako `char meno[] = "Milan"`.
  - každý znak je element pola a alokuje jeden byte pamäte
  - posledný znak je vždy `\0` označujúci koniec reťazca
- Pre lepšiu manipuláciu môžeme použiť knižnicu `<string.h>`.

Praktická ukážka –>

Call by Value

- Kópia hodnoty argumentu je posunutá do funkcie.
- Zmeny hodnoty v rámci funkcie nemajú efekt na pôvodnú premennú (t.j. mimo funkcie).

Call by Reference

- Adresa argumentu je posunutá do funkcie, žiadna hodnota nie je kopírovaná.
- Zmeny vnútri funkcie ovplyvňujú pôvodné premenné.

Call by Value

```
int increment(int x) { -> do funkcie je kópia hodnoty pôvodnej premennej

    x++;  -> inkrement je alokovaný ako nová hodnota vrátená funkciou

}

int main() {

    int num = 5;

    printf("num is now: %d", increment(num));  -> vypíše číslo 6

    return 0;

}
```

Call by Reference

```
void increment(int *x) { -> do funkcie je posunutá adresa pôvodnej premennej

    (*x)++;  -> inkrement je uložený do pôvodnej premennej, podľa adresy pointra

}

int main() {

    int num = 5;

    increment(&num);

    printf("num is now: %d", num);  -> vypíše číslo 6

    return 0;

}
```

Otázky na zamyslenie

- Aký je rozdiel medzi pointerom a premenou?
- Môže existovať pointer na pointer?
  - Ak áno akú má hodnotu?
- Závisí veľkosť pointra na dátový typ pointra?
- Prečo je nutné definovať dátový typ pointra?
- Existuje pointer, ktorý je možné použiť na akýkoľvek dátový typ?

https://zapr.interes.group/exercises/exercise-6