evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

DUM 16 téma: Pointery

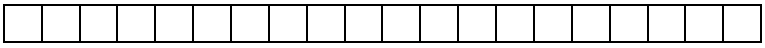
ze sady: 2 **tematický okruh sady:** Vyšší programovací jazyky
ze šablony: 10 – Algoritmizace a programování **určeno pro:** 2. a 3. ročník
vzdělávací obor: 18-20-M/01 Informační technologie
26-41-M/01 Elektrotechnika - Elektronické počítačové systémy
vzdělávací oblast: odborné vzdělávání
metodický list/anotace: VY_32_INOVACE_10216ml.pdf
pomocné soubory: uk1.cpp, uk2.cpp, uk3.cpp, uk4.cpp, cviceni.cpp, cviceni.exe

Doplňte:
Proměnná je .....
Proměnná je určena .....

I. Adresa proměnné

a. Uložení proměnné v paměti PC

```
int cislo;
```



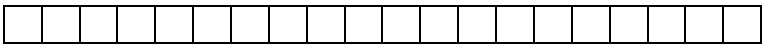
```
int _____
cislo _____
Adresa _____
```

b. Pointer (ukazatel)

Pointer na proměnnou je adresa této proměnné v paměti počítače (je to číslo určující adresu).

```
int *p_cislo;
```

Proměnná `p_cislo` je typu ukazatel na `int` a mohu do ní uložit adresu proměnné typu `int`.

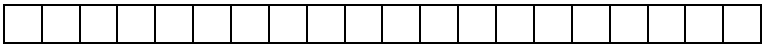


c. Operátor reference - &

Tento operátor mi zjistí adresu proměnné.

```
p_cislo=&cislo;
```

Do proměnné `p_cislo` se uloží adresa proměnné `cislo`.

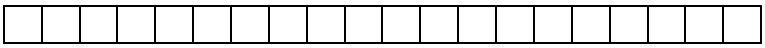


a. Operátor dereference - *

Tento operátor se odkazuje na proměnnou s příslušnou adresou.

```
*p_cislo=10;
```

Do proměnné, která je na adrese uložené v proměnné `p_cislo`, se uloží hodnota 10 (v našem případě se uloží hodnota 10 do proměnné `cislo`).



II. Cvičení 1

Zjednodušeně je: & _____

* _____

a. Zjistěte, co dělají následující příkazy. Předpokládejte: `int i, *p_i;` **Které z nich nedávají smysl nebo jsou syntakticky špatně?**

```
i=3;
*p_i=4;
p_i=3;
i=p_i;
i=*p_i;
i=&p_i;
*p_i=i;
p_i=&i;
```

b. Doplňte tabulku za předpokladu `int i, *p_i1, *p_i2`**.**

	`i` – adresa 10	`p_i1` – adresa 20	`p_i2` – adresa 30
`i=1;`			
`p_i1=&i;`			
`*p_i1=2;`			
`i=*p_i1+1;`			
`p_i2=&i;`			

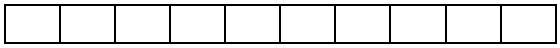
III. Pointer jako parametr funkce

a. Pole a řetězec X pointer

```
char retez[20];
int pole[10];
```

Proměnná `retez` je typu `*char`, a proměnná `pole` je typu `*int`. Jsou to tedy pointery a lze s nimi pracovat jako s pointery.

```
*pole=50;
*(pole+5)=100;
```



Popište a nakreslete, co dělají následující příkazy s pamětí.

b. Paměť programu při volání funkce

Veškeré funkce se ukládají v paměti na zásobníku.
Při volání funkce se hodnoty parametrů zkopírují do volané funkce.

Př. – špatně:
```
int pole[1000];
void bubble(int pole[1000]){…}
```

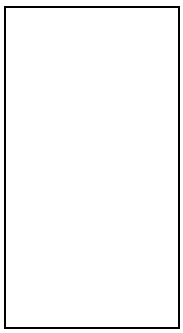
Proč? _____

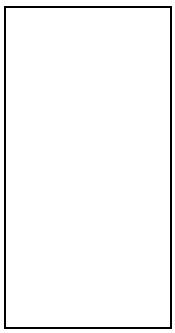
_____

Jaké je správné řešení? _____

_____

_____





```
void fce1(int *x){
  *x=*x+10;
}
```

```
void fce2(int x){
  x=x+10;
}
```

c. Pole a řetězec jako argument funkce

Pole čísel/řetězec je ukazatel na `int/char`. Mohu tedy předávat ne celé pole, ale pouze pointer na pole. To znamená adresu v paměti počítače, kde toto pole začíná!

Hlavnička funkce: `void bubble(int *pole){…}`

Volání funkce:

```
int pole[1000];
bubble(pole);
```

IV. Cvičení

a. Najděte chybu:

```
1) void bubble(int pole[10]){…}
2) void fce(int *x){…}
   int cislo;
   fce(*cislo);
3) int fce(int *x){
     return x/2;
   }
4) int fce(int *a, int *b){
     *a=(*a)*(*a);
     *b=(*b)*(*b);
     return *a+*b;
   }
   …
   int x;
   x=fce(&x,&x);
```

b. Co dělají následující funkce:

```
1) int fce1(char *x){
     int i=0;
     while(x[i]!='\0')
       i++;
     return i;
   }
2) void fce2(int *x, int y){
     int i;
     for(i=0;i<y;i++)
       x[i]=0;
   }
```

c. Naprogramujete:

1) Vytvořte funkci `vymen(…)`, která má na vstupu odkaz na dvě čísla a tato funkce prohodí hodnoty těchto proměnných.

2) Vytvořte funkci `bubble(…)`, která dostane jako parametr pole a počet prvků a tato funkce setřídí toto pole.

3) Vytvořte funkci `navelka(…)`, která dostane na vstupu řetězec a změní všechna malá písmenka v tomto řetězci na velká.