FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ústav teoretické a experimentální elektrotechniky

Seminář C++

2. přednáška

Autor: doc. Ing. Jan Mikulka, Ph.D.

2023

Obsah přednášky

- Dynamická alokace paměti
- Vícerozměrná pole, pointery
- Vracení hodnoty parametrem
- Příklady k řešení

Vícerozměrné pole

- 1D: vektor hodnot
- 2D: matice hodnot
- 3D, 4D, 5D, …

- Textový řetězec – 1D pole znaků

Dynamická alokace paměti

- C

```
void* malloc (size_t size);
```

```
void free (void* ptr);
```

- C++

```
pointer = new type [number_of_elements]
```

```
delete pointer;
delete[] pointer;
```

Dynamická alokace paměti

```c
int _tmain(int argc, _TCHAR* argv[])
{
    int i,n;
    char * buffer;

    printf ("Jak dlouhy retezec? ");
    scanf ("%d", &i);

    buffer = (char*) malloc (i+1);
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Nahodny retezec: %s\n",buffer);
    free (buffer);

    system("pause");
    return 0;
}
```

```c
int _tmain(int argc, _TCHAR* argv[])
{
    int i,n;
    char * buffer;

    printf ("Jak dlouhy retezec? ");
    scanf ("%d", &i);

    buffer = new char[i+1];
    if (buffer==NULL) exit (1);

    for (n=0; n<i; n++)
        buffer[n]=rand()%26+'a';
    buffer[i]='\0';

    printf ("Nahodny retezec: %s\n",buffer);
    delete[] buffer;

    system("pause");
    return 0;
}
```

Dynamická alokace paměti

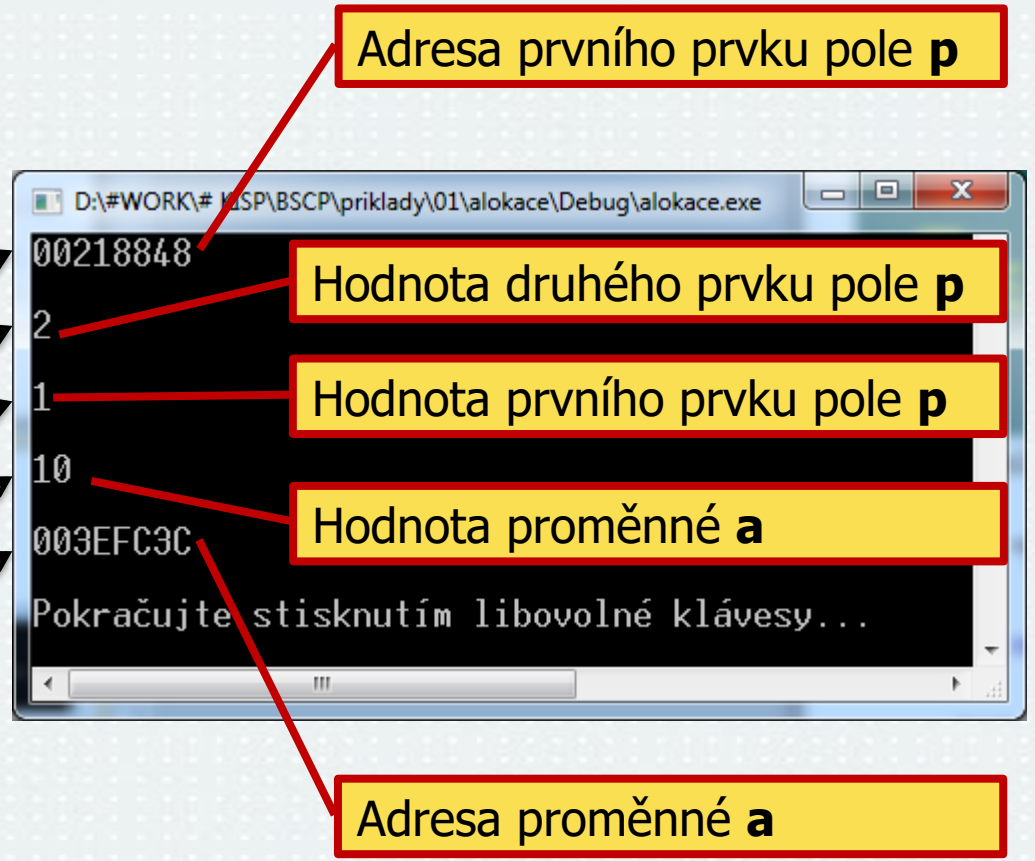
```
int *p;

p = new int(3);

p = new int[3];
```

```
delete p;

delete [] p;
```

- (n) – inicializace hodnotou n
- [n] – alokace pole o velikosti n

Pointery

```
int _tmain(int argc, _TCHAR* argv[])
{
    int *p, a;

    p = new int[3];

    p[0] = 1;
    p[1] = 2;
    p[2] = 3;

    cout << p << endl;
    cout << p[1] << endl;
    cout << *p << endl;

    a = 10;

    cout << a << endl;
    cout << &a << endl;

    system("pause");
    return 0;
}
```



Nulou ukončené řetězce

- 1D pole znaků typu **char**

- Aby bylo poznat, jak je řetězec dlouhý, je ukončen znakem '\0'

```
for (n=0; n<i; n++)
    buffer[n]=rand()%26+'a';
buffer[i]='\0';
```

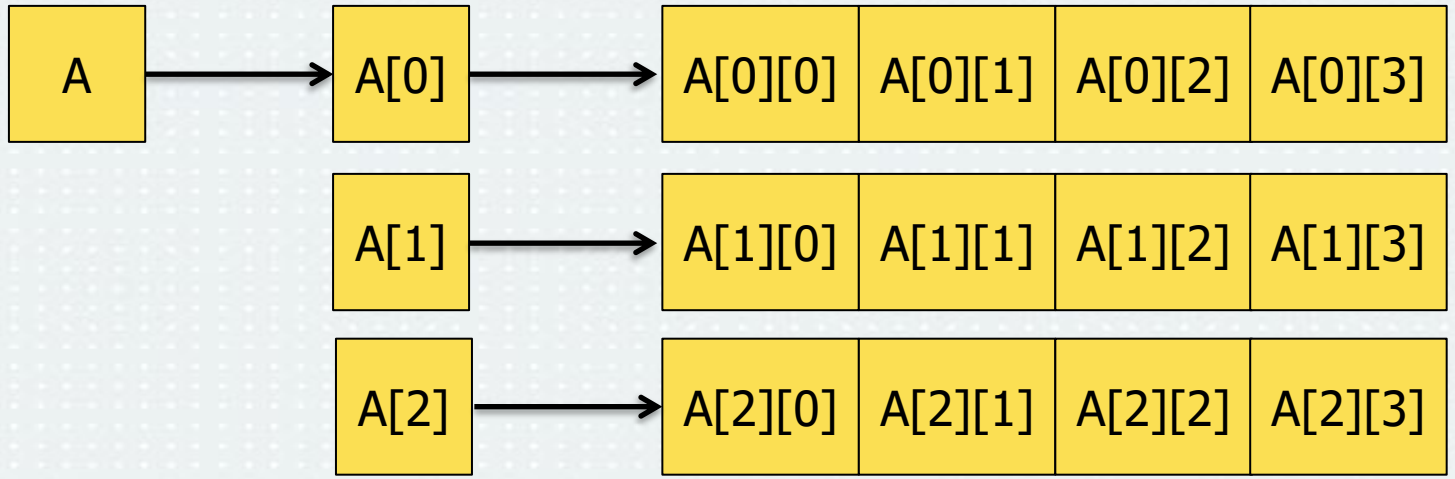
Dynamická alokace ve 2D

- Dále budeme používat pouze operátor **new** a **delete** (OOP)
- Pro alokaci 1D pole stačí 1x volat **new**,
  pro dealokaci 1x **delete**
- Práce s 2D polem je složitější, ale i xD pole lze převést na 1D pole

Dynamická alokace ve 2D

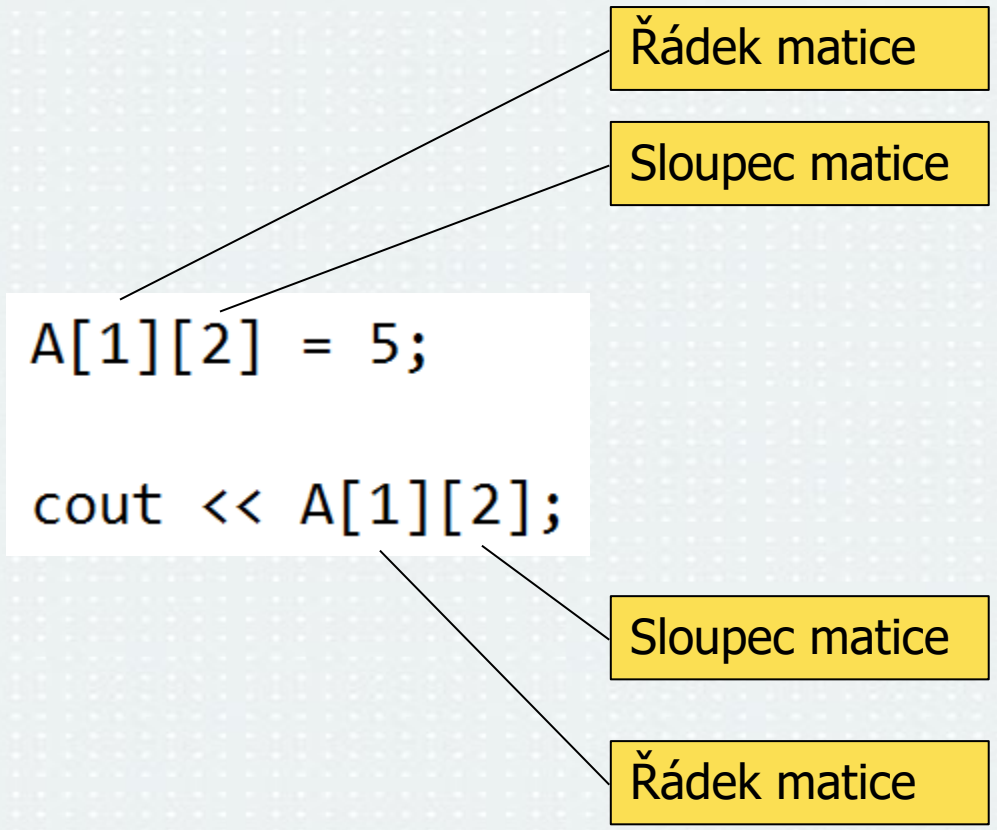
```
int **A;

A = new int*[rows];
for (int i = 0; i < rows; i++)
    A[i] = new int[cols];
```

```
for (int i = 0; i < rows; i++)
    delete[] A[i];
delete[] A;
```



Přístup k prvkům 2D pole

```
int **A;

A = new int*[rows];
for (int i = 0; i < rows; i++)
    A[i] = new int[cols];
```



Předávání pole parametrem funkce

```
void napln(int **X)
{
    for (int r = 0; r < rows; r++)
        for (int c = 0; c < cols; c++)
            X[r][c] = rand()%10;
}

void tiskni(int **X)
{
    for (int r = 0; r < rows; r++)
    {
        for (int c = 0; c < cols; c++)
            cout << X[r][c] << " ";
        cout << endl;
    }
}
```

```
int _tmain(int argc, _TCHAR* argv[])
{
    int **A;

    A = new int*[rows];
    for (int i = 0; i < rows; i++)
        A[i] = new int[cols];

    napln(A);
    tiskni(A);

    for (int i = 0; i < rows; i++)
        delete[] A[i];
    delete[] A;

    system("pause");
    return 0;
}
```

Vracení pole parametrem

- V případě, že potřebujeme vrátit větší množství dat. U objektů výrazně urychluje běh programu a šetří paměť.

- V případě, že potřebujeme nastavit hodnotu proměnné v nadřazené funkci

```
void soucet(int **A, int **B, int **C)
{
    for (int r = 0; r < rows; r++)
        for (int c = 0; c < cols; c++)
            C[r][c] = A[r][c] + B[r][c];
}
```

```
int **A, **B, **C;
soucet(A, B, C);
```

Děkuji Vám
za pozornost.

mikulka@vut.cz
www.utee.fekt.vut.cz