Lecture 3: Pointers and functions

Johannes Gerstmayr and Markus Walzthöni

Material: Jonas Kusch and Martina Prugger
University of Innsbruck

October 16, 2023

Last goals: You are able to

- ☑ use conditional statements
- ☑ use loops
- ☑ understand memory management
- ☑ start to use pointers

Today's learning goals: You will be able to

- ☐ generate dynamic and static arrays
- ☐ understand pointer arithmetics
- ☐ use functions

Last goals: You are able to

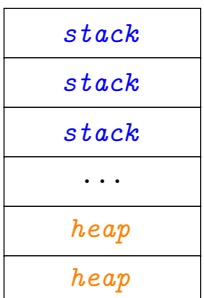
- ☑ use conditional statements
- ☑ use loops
- ☑ understand memory management
- ☑ start to use pointers

Today's learning goals: You will be able to

- ☐ generate dynamic and static arrays
- ☐ understand pointer arithmetics
- ☐ use functions

Why?

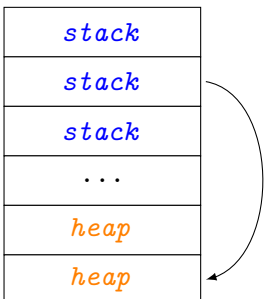
- Pointers give control over memory.
- Pass data to a different program part without copying it (`functions`,`classes`,...).
- Control when to delete data (dynamic vs. static memory).



- **static** memory managed by compiler (*stack*)
- **dynamic** memory managed by user (*heap*)
- dynamic memory can be accessed with pointers (stored in stack)
- address space in heap is accessed with `new`

Why?

- Pointers give control over memory.
- Pass data to a different program part without copying it (`functions`,`classes`,...).
- Control when to delete data (dynamic vs. static memory).



- **static** memory managed by compiler (*stack*)
- **dynamic** memory managed by user (*heap*)
- dynamic memory can be accessed with pointers (stored in stack)
- address space in heap is accessed with `new`

Code presentation

Code presentation

```
#include <iostream>

int main(){
    double dStatic = 0.1;
    double *dDynamic = new double; // allocate memory in heap
    *dDynamic = 0.1;

    std::cout<<dStatic<<" "<<*dDynamic<<std::endl;

    delete dDynamic; // free memory

    std::cout<<dStatic<<" "<<*dDynamic<<std::endl;

    return 0;
}
```

Arrays in heap

Arrays in heap

```
#include <iostream>

int main(){
    double *v = new double [2]; // allocate array of size 2 in heap
    v[0] = 0.1;
    v[1] = 0.12;

    delete [] v; // free memory of entire array

    return 0;
}
```

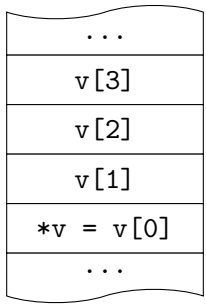
Arrays in heap

```cpp
#include <iostream>

int main(){
    double *v = new double [2]; // allocate array of size 2 in heap
    v[0] = 0.1;
    v[1] = 0.12;

    delete [] v; // free memory of entire array

    return 0;
}
```

Task

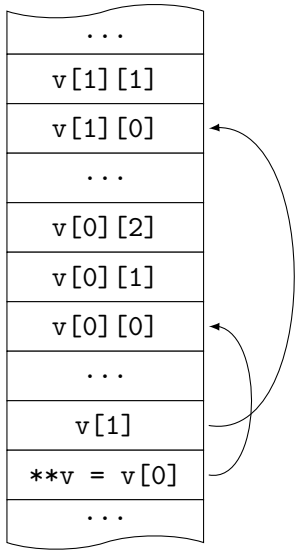
Rewrite your ODE solver by using dynamic arrays. Make sure to free your memory before the program terminates.

Arrays in heap



- We store address of `v[0]` on `v`.
- Since `v[1]`,... neighbor `v[0]` we also know their addresses.
- More about this when talking about pointer arithmetics.

Multi-dimensional arrays in heap



Multi-dimensional arrays in heap

Multi-dimensional arrays in heap

```
#include <iostream>

int main(){
    int n = 3, m = 4;
    double** v = new double* [n]; // allocate array of pointers

    for( long i = 0; i < n; ++i)
        v[i] = new double [m]; // allocate double array for every v[i]

    v[0][1] = 0.1;

    for( long i = 0; i < n; ++i)
        delete [] v[i]; // delete array of doubles for every v[i]

    delete [] v; // delete array of pointers

    return 0;
}
```

Multi-dimensional arrays in heap

Task

Implement a 3-dimensional array a with dimension $n_1 = 2, n_2 = 3, n_3 = 4$. Fill the array with numbers $a_{ijk} = i + j + k$. Do not forget to free your memory before the program terminates.

What does the code do? What happens in memory?

```
#include <iostream>

int main(){
    double* d = new double;
    *d = 0.1;
    double* p = d;

    delete p;

    std::cout<<*d<<std::endl;

    return 0;
}
```

What does the code do? What happens in memory?

```
#include <iostream>

int main(){
    bool condition = true;

    if( condition ){
        double* d = new double;
        *d = 0.1;
    }

    std::cout<<*d<<std::endl;

    return 0;
}
```

What does the code do? What happens in memory?

```
#include <iostream>

int main(){
    bool condition = true;
    double* d;

    if( condition ){
        d = new double;
        *d = 0.1;
    }

    std::cout<<*d<<std::endl;

    return 0;
}
```

Last goals: You are able to

- ☑ use conditional statements
- ☑ use loops
- ☑ understand memory management
- ☑ start to use pointers

Today's learning goals: You will be able to

- ☑ generate dynamic and static arrays
- ☐ understand pointer arithmetics
- ☐ use functions

Last goals: You are able to

- ☑ use conditional statements
- ☑ use loops
- ☑ understand memory management
- ☑ start to use pointers

Today's learning goals: You will be able to

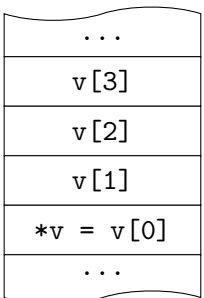
- ☑ generate dynamic and static arrays
- ☐ understand pointer arithmetics
- ☐ use functions

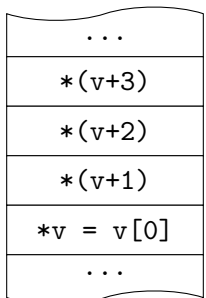
Pointer arithmetics

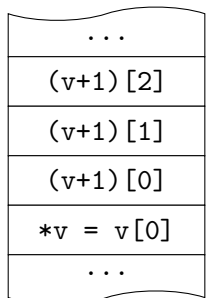
- Arithmetics on pointers allowed.
- d[i] equivalent to = *(d + i)

```
#include <iostream>

int main(){
    double* d;
    d = new double [4];
    d[0] = 0.0; d[1] = 0.1; d[2] = 0.2;

    std::cout<< *d << " " << *(d + 1) <<std::endl;

    return 0;
}
```

Pointer arithmetics



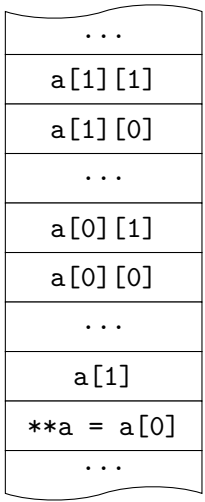




What's the output?

```
#include <iostream>

int main(){
    long** a = new long* [2];
    a[0] = new long [2];
    a[1] = new long [2];

    for( long i = 0; i < 2; ++i )
        for( long j = 0; j < 2; ++j )
            a[i][j] = i + j;

    std::cout<< *(a[1]+1) << " " << (*a - 1)[2] <<std::endl;
    std::cout<< *((a + 1)[0] + 1) <<std::endl;
    std::cout<< (a + 2)[0][2] << std::endl;

    return 0;
}
```

What's the output?



→ `*(a[1]+1)`

→ `(*a - 1)[2]`

→ `*((a + 1)[0] + 1)`

→ `(a + 2)[0][2]`

Last goals: You are able to

- ☑ use conditional statements
- ☑ use loops
- ☑ understand memory management
- ☑ start to use pointers

Today's learning goals: You will be able to

- ☑ generate dynamic and static arrays
- ☑ understand pointer arithmetics
- ☐ use functions

Last goals: You are able to

- ☑ use conditional statements
- ☑ use loops
- ☑ understand memory management
- ☑ start to use pointers

Today's learning goals: You will be able to

- ☑ generate dynamic and static arrays
- ☑ understand pointer arithmetics
- ☐ use functions

Functions

- Is everyone familiar with functions in programming languages?

```
<return_data_type> function_name( <input_1>, <input_2>,... ){
    \\ function body
    return <return_value>
}
```

Functions

- Is everyone familiar with functions in programming languages?

```
<return_data_type> function_name( <input_1>, <input_2>,... ){
    \\ function body
    return <return_value>
}
```

```cpp
#include <iostream>

double add(double a, double b){
    double c = a + b;
    return c;
}

int main(){
    std::cout << add(1,2) <<std::endl;
    return 0;
}
```

Your turn

Task

Rewrite your ODE solver as a function which takes start time and time grid as input and returns the solution at each time point as output. Use another function to define the right hand side of your ODE.