Loeng 17.09

RISTO HEINSAR

RISTO.HEINSAR [ÄT] TTU.EE

BLUE.PRI.EE/TTU

Moodle

Registreerige end kursusele:
https://moodle.taltech.ee/course/view.php?id=3347

Moodle kasutuse ulatuse täpsustab praktikumi õppejõud.

Teemad

Algoritmid

UML

C keel

Massiivid

Näited, näited, näited

Algoritm

Eeskiri jõudmaks algolekust soovitud sihini

Levinud matemaatika ja IT valdkondades

- Kuidas leida aritmeetilist keskmist
- Kuidas lahendad lineaarvõrrandisüsteemi
- Kuidas leida jada suurimat liiget
- Kuidas sorteerida arvujada

Sisuliselt võib algoritmiks kirjutada kõike

- Kuidas valmistada pannkooke
- Kuidas panna kokku sektsioonkappi
- Kuidas vahetada piduriklotse
- Kuidas teostada isiku taustakontroll enne laenulepingut

Olulised omadused

Võimalikult täpne - puuduvad ebamäärased tegevused

- Tegevus: pane särk selga
- Mis tüüpi särk?
- Milliste omadustega särk?
- Mis pidi?
- Kelle selga?

Üheselt mõistetav

- Nt personaliosakond ja IT osakond peavad saama üheselt aru protsessidest, mis viiakse läbi töötaja vallandamisel

UML

Unified modelling language

Kasutatakse peamiselt tarkvarasüsteemide spetsifitseerimiseks, visualiseerimiseks, arenduseks ja dokumenteerimiseks

20 aastat keelearendust

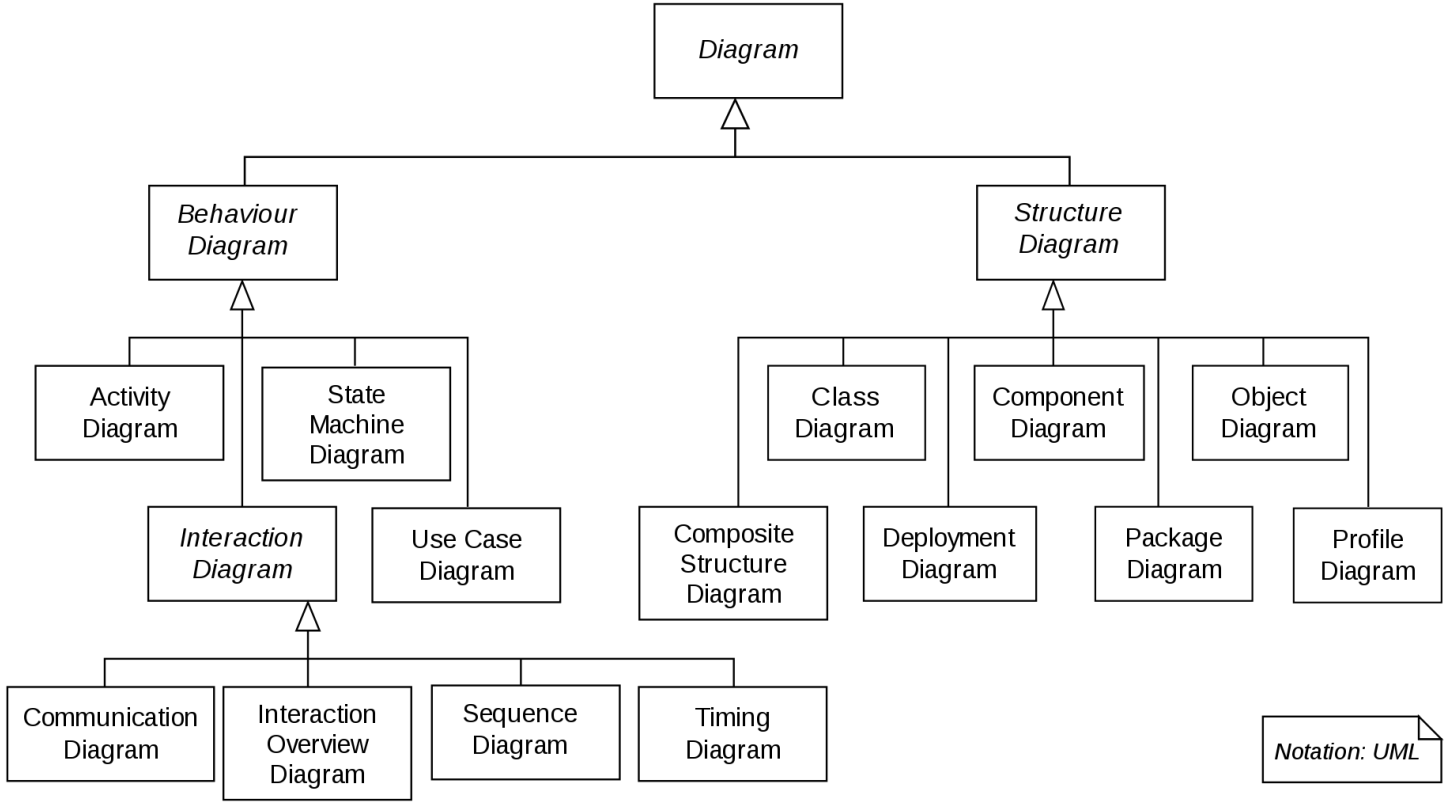
- UML 2.5.1 – detsember 2017

Programmeerimiskeelest ja arendusprotsessidest sõltumatu

Sobib ka reaalse maailma nähtuste kirjeldamiseks

https://www.omg.org/spec/UML/2.5.1/PDF (796 lk)

UML diagrammid



UML tegevusdiagrammid

Kirjeldab tegevuste jada

Näitame tegevusi, hargnemisi ja koondumisi, paralleelseid toiminguid

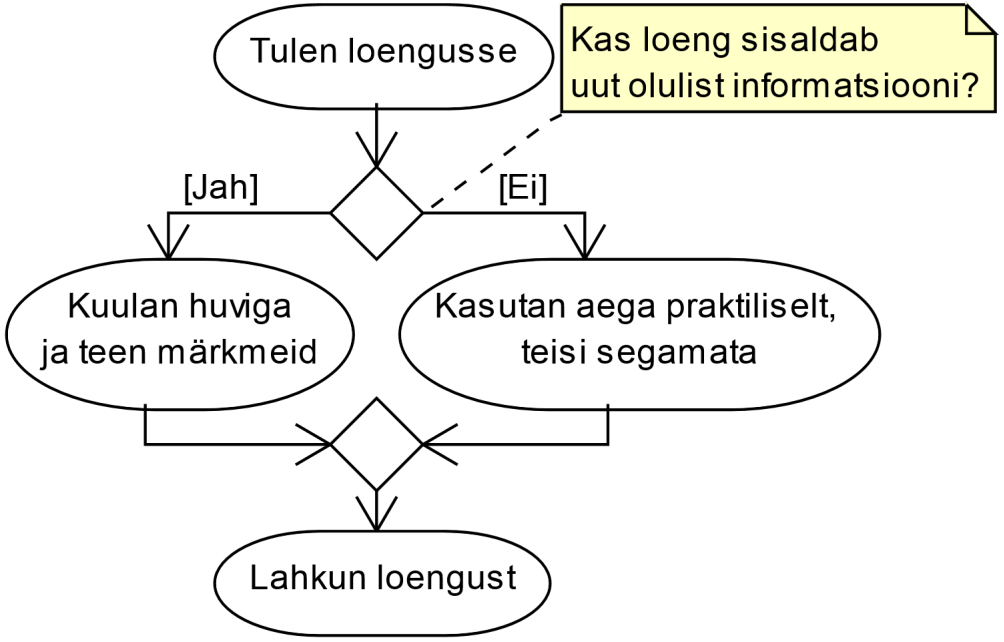
Alati 1 algolek, võimalik mitu lõppolekut

Tükeldame loogilisteks plokkideks kasutades ujumisradu (*swimlane*)

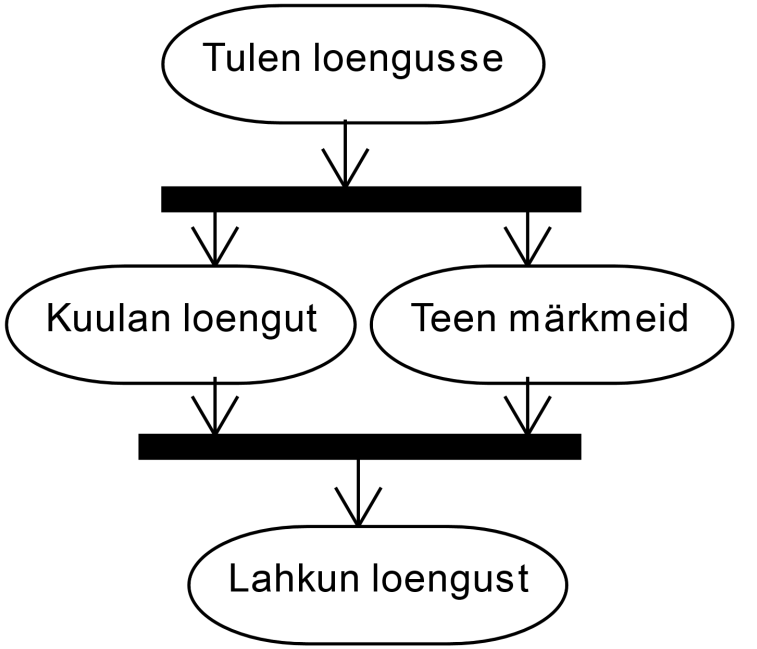
Igast tegevusest **peab olema võimalik** jõuda vähemalt ühte lõppolekusse

Tingimus vs paralleeltegevus

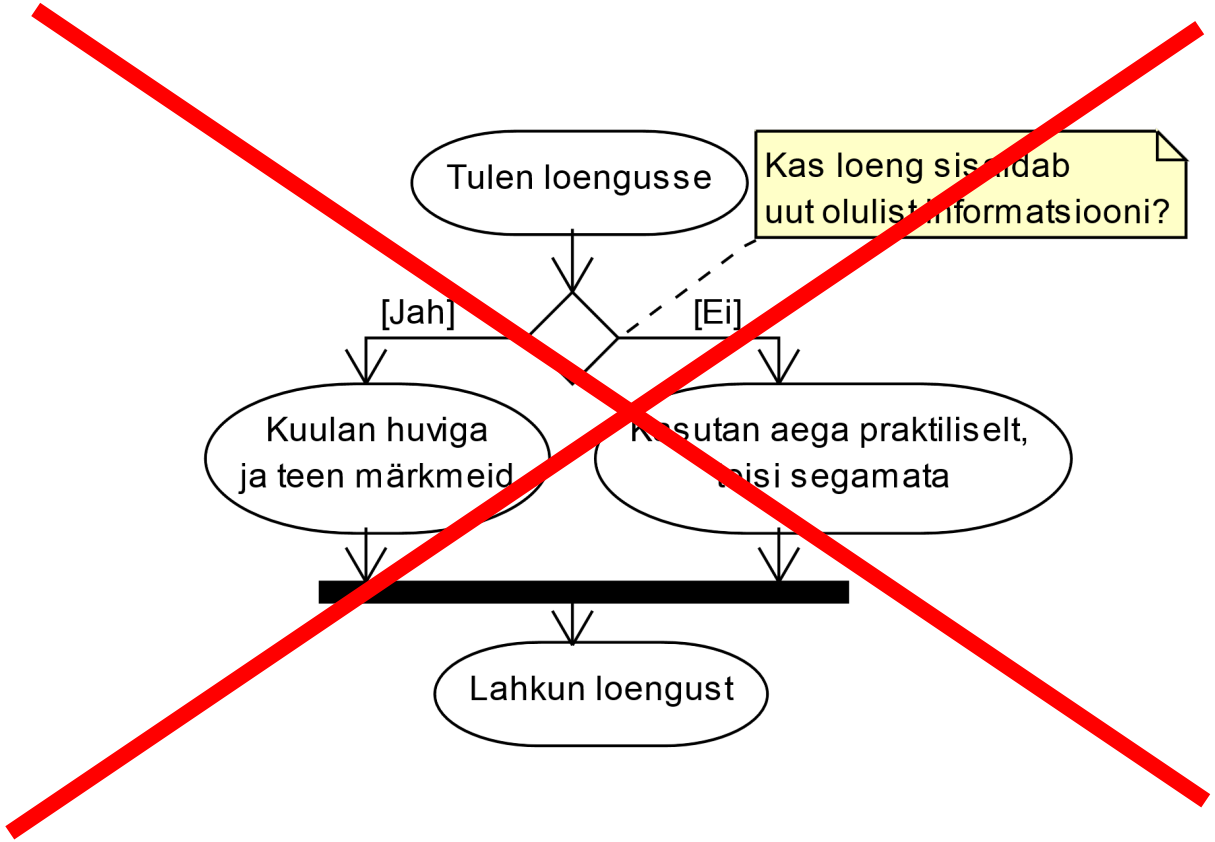
TINGIMUS



PARALLEELTEGEVUS



Sünkronisatsiooniribaga võib ainult paralleeltegevust lõpetada



C keel

Sai alguse 1972

Inspiratsiooniks suurele osale programmeerimiskeeltest

Objektorienteeritud versioon C++

TIOBE indeksi alusel #2 keel maailmas

Laiatarbekeel (*General purpose language*)

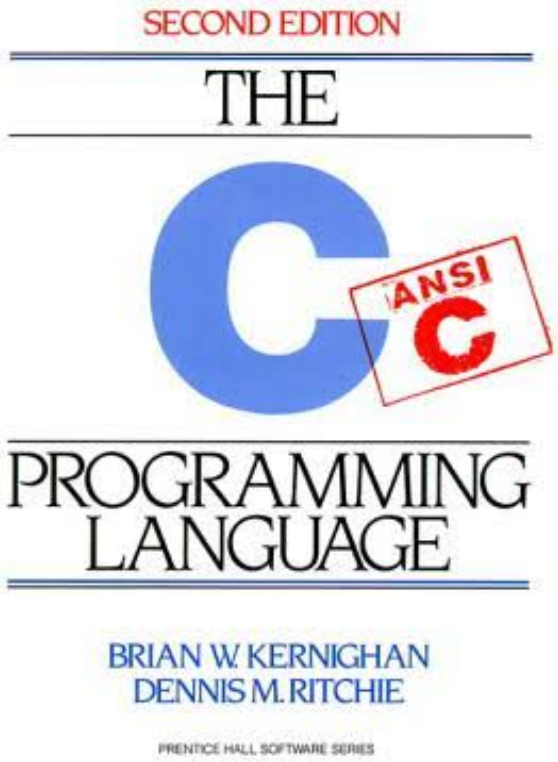
Lähtekoodist kompileeritav keel

Töötab peaaegu igal seadmel

„Maagiavaba“

Primitiivne aga kiire

Jätkuvalt arenduses



Kasutusvaldkonnad

Sardsüsteemid (*Embedded systems*)

Tüürlid/ajurid (*Hardware drivers*)

Operatsioonisüsteemide tuumad (*Operating system kernels* - Linux, Windows, Mac OS, Android, iOS)

Veebiserverid (*Web servers* - nginx, apache)

Andmebaasisüsteemid (*Database management systems* - Oracle, MySQL, PostgreSQL, MariaDB)

Programmeerimiskeelte moodulid (nt pythoni moodulid)

Jne.

Lihtsamad andmetüübid

Andmed	Andmetüüp	Formaat	Näited
Täisarvud	int	%d	1, 3, -55, 0
Murdarvud	Float double	%f, %g %lf, %lg	1.2, -5.5, 1.0, 0.0, -0.0, 5.2e3, 10.332f
Tähemärgid	char	%c	‘a’, ‘x’, ‘-’, ‘ ‘
Sõned	char []	%s	„Hello“, „Lorem ipsum“
Tõeväärtus	bool	%d	True (1), false (0)

https://en.wikipedia.org/wiki/C_data_types

Veel andmetüüpidest

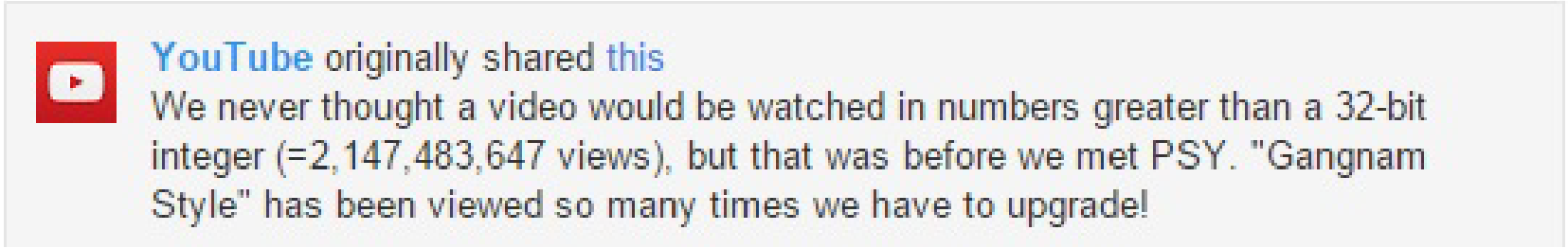
Andmetüüp tuleb määrata muutuja deklareerimise hetkel

Selle järgi eraldatakse kindel kogus mälu

Andmetüübist sõltub esitustäpsus ja -ulatus

- Märgiga 2-baidine täisarv: [−32,767, +32,767]
- Märgita 2-baidine täisarv: [0, 65,535]
- Märgiga 4-baidine täisarv: [−2,147,483,647, +2,147,483,647]
- Märgiga 8-baidine täisarv: [−9,223,372,036,854,775,807, +9,223,372,036,854,775,807]

Probleemid: ületäitumine (*overflow*), murdarvud ebatäpsed (*floating point precision*)



Maagilised numbrid

Väärtused, millel puudub kergesti arusaadav tähendus

3.14159

15.6466

8.31462618

PK

ABADBABE

https://en.wikipedia.org/wiki/Magic_number_(programming)

#define makrod (preprotsessori direktiiv)

#define makrod täidetakse **enne koodi kompileerimist**

Töötab suuresti *„find and replace“* ehk asendamise põhimõttel

Paigutatakse koodi algusse kohe pärast #include käske

```
#define <tunnus> <väärtus>
```

Eesmärk: Likvideerida koodist maagilised numbrid, konstandid

Erinevalt muutujatest ei saa nende väärtusi programmi keskel muuta!

Koodimisstiil: suurtähed, sõnade vahel allkriips

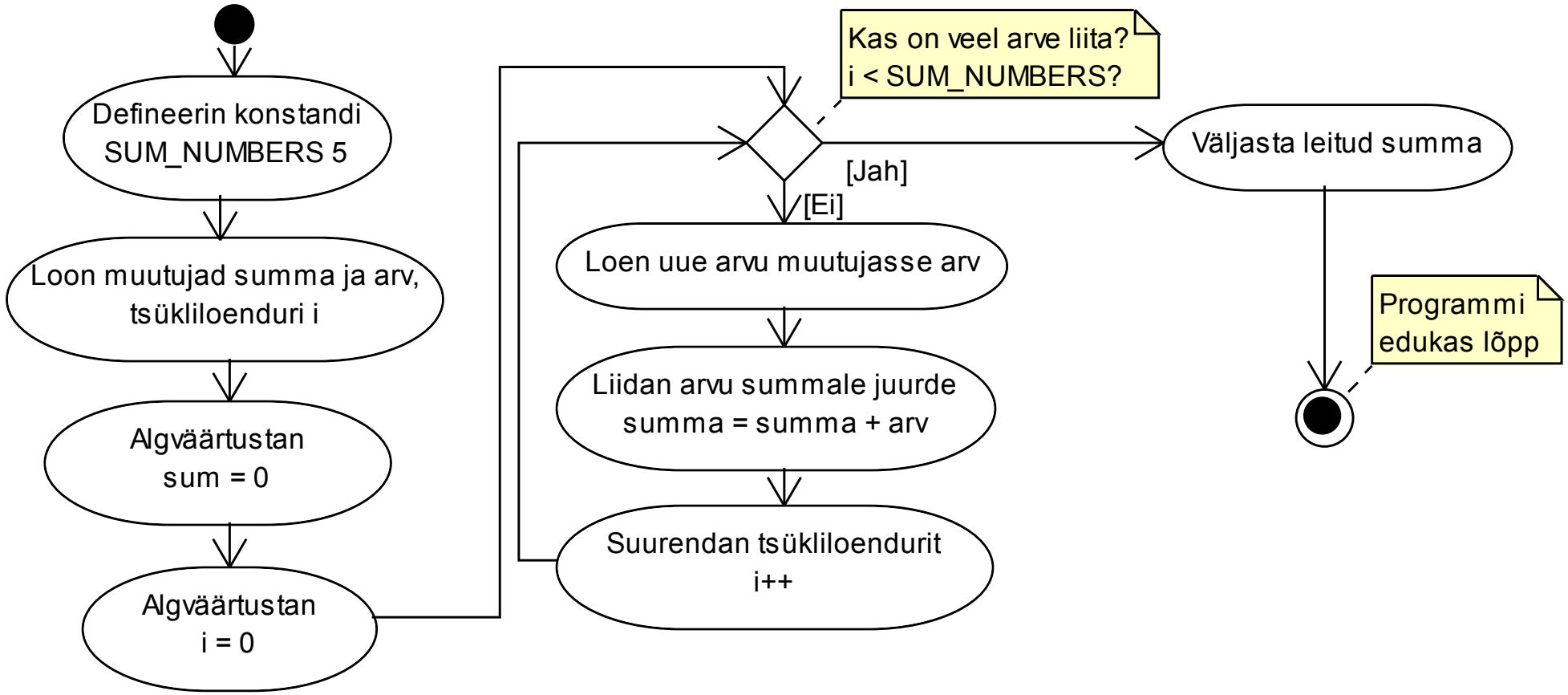
Maagilised numbrid

```
#define SIZE 7
#define TEXTFIELD_LEN 30
#define MATH_PI 3.14159
#define EEK_EUR_CONV_RATE 15.6466
#define UNIVERSAL_GAS_CONSTANT 8.31462618
#define OPTIONS_FILE "settings.ini"
#define ZIP_FILE_HEADER "PK"
#define APPLE_BOOT_ZERO_BLOCK "ABADBABE"
```

Näide: viie arvu liitmine C

```c
#include <stdio.h>

#define SUM_NUMBERS 5

int main(void)
{
    int i, inputNum;
    int sum = 0;

    for (i = 0; i < SUM_NUMBERS; i++)
    {
        printf("Enter num %d out of %d: ", i + 1, SUM_NUMBERS);
        scanf("%d", &inputNum);
        sum += inputNum;
        printf("The sum is now %d\n", sum);
    }
    return 0;
}
```

Näide: viie arvu liitmine UML



Massiivid

Massiiv on indekseeritud ühetaoliste andmete kogum

- St täisarvud, murdarvud, tähemärgid, sõned, …

Massiivi liikmete poole pöördume **indeksi** abil

Massiivi elementide indekseerimine hakkab **nullindast** (enamikes keeltes)

- St viieliikmelise massiivi elementide indeksid on 0 – 4

Massiivist väljaspoole indekseerida **ei tohi**! (*out of bounds*)

1-dimensioonilist massiivi võidakse kutsuda ka **vektoriks** või **jadaks**

2-dimensioonilist massiivi võidakse kutsuda **maatriksiks**

- Kujuta ette malelauda, Exceli tabelit või kordinaatteljestikku
- Erijuht: kui mõlemad dimensioonid on samapikad, siis on see **ruutmaatriks**

Massiivi deklareerimine

Massiivi suurus määratakse ära muutuja deklareerimise hetkel

Muutuja nime järele lisatakse nurksulgudes pikkus

```
int arvud[5]; // massiiv 5 täisarvu jaoks
char tekstiVali[15]; // massiiv 14+1 tähemärki
```

Massiivi on võimalik algväärtustada deklareerimise hetkel

```
int arvud[5] = {0}; // algväärtustab kõik liikmed nullideks
int arvud[5] = {0, -5, 3, 12, 3};
int arvud[10] = {0, -5, 3, 12, 3}; // vabad kohad lõpus
int arvud[] = {0, -5, 3, 12, 3}; // suurus automaatne (1-dim!)
```

1-dimensioonilise massiivi kasutamine

Massiivi deklareerimine

```
int arvud[5];
```

Massiivi väärtuste poole pöördume koos vastava elemendi indeksiga

```
arvud[indeks];
```

1-dimensioonilise massiivi kasutamine

Kõik operatsioonid on samamoodi nagu üksiku muutujaga, lisandub vaid indeks

Loeme klaviatuurilt kolmanda elemendi:

```
scanf("%d", &arvud[2]);
```

Trükime välja massiivi kolmanda elemendi:

```
printf("%d", arvud[2]);
```

Määrame käsitsi kolmanda elemendi:

```
arvud[2] = 93;
```

Liidame summale massiivi kolmanda elemendi:

```
summa = summa + arvud[2];
```

Näide: Massiivid ilma tsükliteta

```c
#include <stdio.h>

#define NUM_COUNT 3

int main(void)
{
    int scoreTable[NUM_COUNT];
    scoreTable[0] = 93;
    scoreTable[1] = 85;
    scoreTable[2] = 51;
    printf("The students achieved the following results: %d, %d and %d\n",
           scoreTable[0], scoreTable[1], scoreTable[2]);
    return 0;
}
```

Massiivi indekseerimine tsüklite abiga

Tsüklite juures rääkisime tsükliloenduritest

- Sedasi saime loendada mitmendat korda tsükkel käib
- Tegu oli täisarvulise muutujaga
- Tegevused: inkrementeerimine või dekrementeerimine 1 võrra
- Olulisel kohal oli algväärtustamine ja tingimus kaua tsükkel kesta võib (mitmeni loendame?)

Antud näite puhul on i täisarvuline muutuja, mis omandab erinevatel iteratsioonidel vastavalt väärtused 0, 1, 2, 3 ja 4

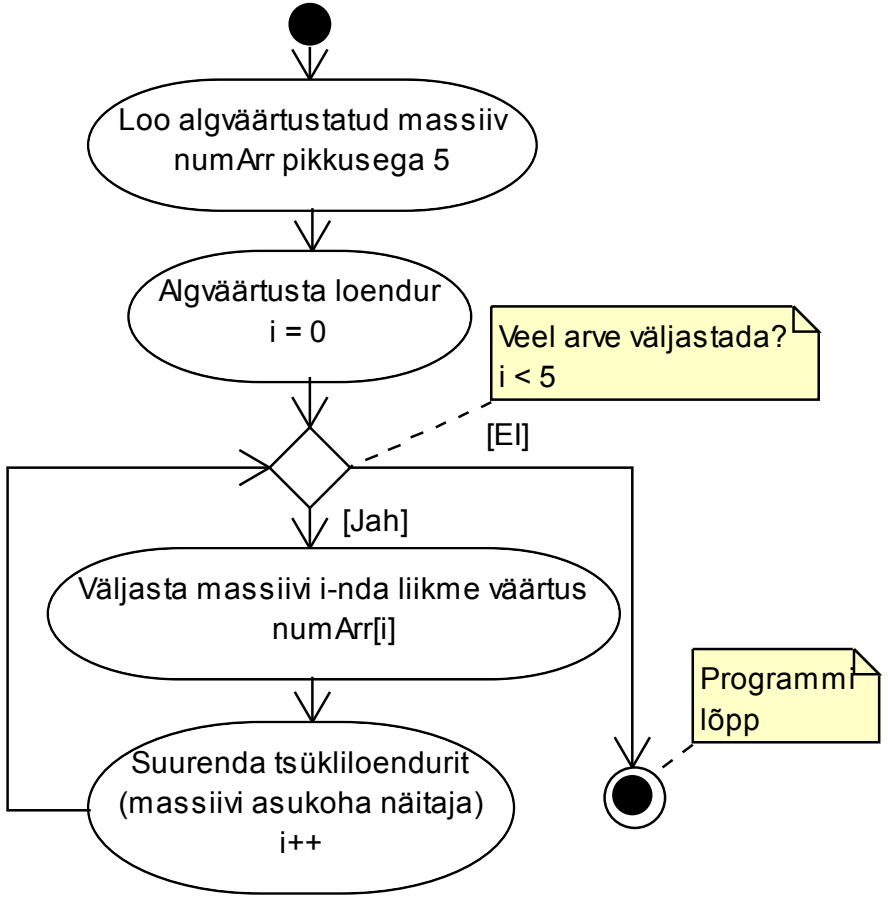
```
for (i = 0; i < 5; i++)
{
    printf("%d\n", i);
}
```

Näide: Massiivid ja tsüklid

```c
#include <stdio.h>

#define NUM_COUNT 5

int main(void)
{
    int numArr[NUM_COUNT] = {19, 2, -5, 133, 0};
    int i;

    printf("The numbers stored in the array are:\n");
    for (i = 0; i < NUM_COUNT; i++)
    {
        printf("%d\n", numArr[i]);
    }
    return 0;
}
```

19	2	-5	133	0
numArr[0]	numArr[1]	numArr[2]	numArr[3]	numArr[4]

Näide UMLis


Loo algväärtustatud massiiv numArr pikkusega 5
Algväärtusta loendur i = 0
Veel arve väljastada? i < 5
[Ei]
[Jah]
Väljasta massiivi i-nda liikme väärtus numArr[i]
Suurenda tsükliloendurit (massiivi asukoha näitaja) i++
Programmi lõpp

Indeksitega manipuleerimine

Adresseerimine indeksi abil liikmed 0 .. NUM_COUNT

```
for (i = 0; i < NUM_COUNT; i++)
```

Adresseerimine indeksi abil liikmed 1 .. 3

```
for (i = 1; i < 4; i++)
```

Adresseerimine indeksi abil liikmed 1 .. NUM_COUNT - 1

```
for (i = 1; i < NUM_COUNT - 1; i++)
```

Adresseerimine indeksi abil liikmed 4 .. 0

```
for (i = NUM_COUNT - 1; i >= 0; i--)
```

Adresseerimine indeksi abil liikmed 0, 2, 4

```
for (i = 0; i < NUM_COUNT; i += 2)
```

Näide: Sisendi piiramine

```c
#include <stdio.h>

#define NUM_COUNT 5

int main(void)
{
    int numArr[NUM_COUNT], i;

    for (i = 0; i < NUM_COUNT; i++)
    {
        do
        {
            printf("Enter num %d / %d: ", i + 1, NUM_COUNT);
            scanf("%d", &numArr[i]);
        }
        while (numArr[i] < 0 || numArr[i] > 9);
        printf("Number %d accepted\n", numArr[i]);
    }
    return 0;
}
```

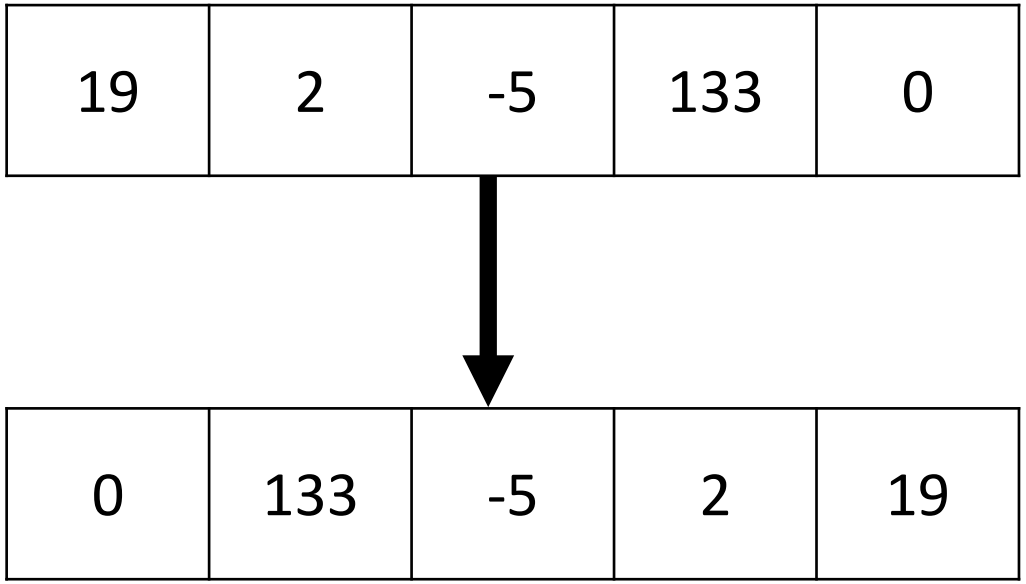
Ülesanne

On antud N liikmeline algväärtustatud täisarvudest koosnev massiiv. Loo uus massiiv, mille liikmeteks oleksid esimese massiivi liikmed, kuid vastupidises järjekorras.

Mõttekohad:

- Kust tuleb N?
- Kust tulevad arvud?
- Mis tüüpi arvudega tegu on?
- Eksisteerib vajadus kahe massiivi järgi
- Kui pikk on teine massiiv?
- Kuidas indekseerida teist massiivi?
- Mis sammuga ja millal indeksid suurenevad?

Lahendusi on rohkem kui üks.



Lahendus 1

```c
#include <stdio.h>

#define NUM_COUNT 5

int main(void)
{
    int numArr[NUM_COUNT] = {1, 3, 5, 7, 9};
    int newArr[NUM_COUNT], i;

    for (i = 0; i < NUM_COUNT; i++)
    {
        newArr[NUM_COUNT - i - 1] = numArr[i];
    }
    return 0;
}
```

Lahendus 2

```c
#include <stdio.h>

#define NUM_COUNT 5

int main(void)
{
    int numArr[NUM_COUNT] = {1, 3, 5, 7, 9};
    int newArr[NUM_COUNT], i, j;

    j = NUM_COUNT - 1;
    for (i = 0; i < NUM_COUNT; i++)
    {
        newArr[j] = numArr[i];
        j--;
    }
    return 0;
}
```

Lahendus 2 kompaktsemalt

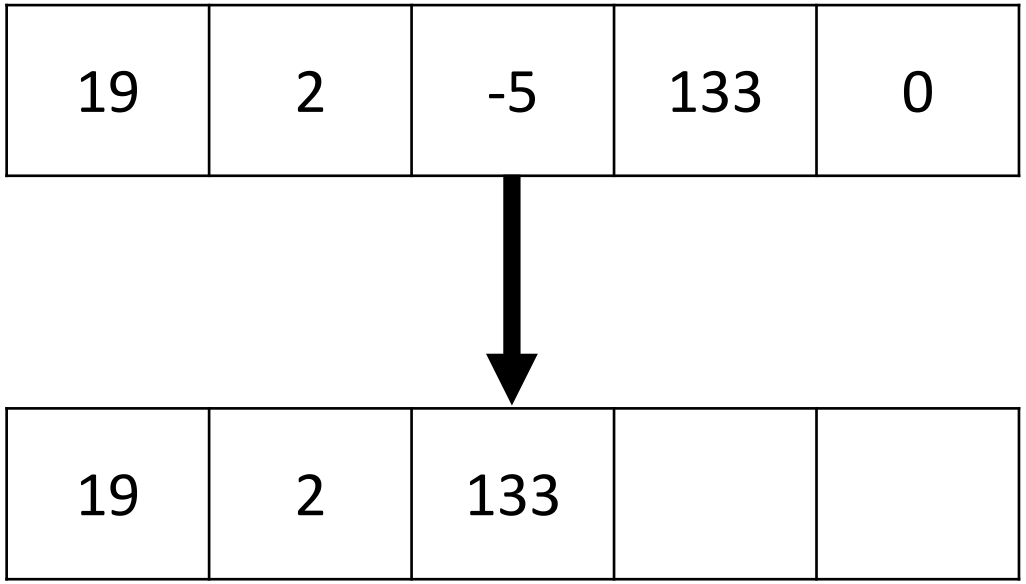
```c
#include <stdio.h>

#define NUM_COUNT 5

int main(void)
{
    int numArr[NUM_COUNT] = {1, 3, 5, 7, 9};
    int newArr[NUM_COUNT], i, j;

    for (i = 0, j = NUM_COUNT - 1; i < NUM_COUNT; i++, j--)
    {
        newArr[j] = numArr[i];
    }
    return 0;
}
```

Ülesanne

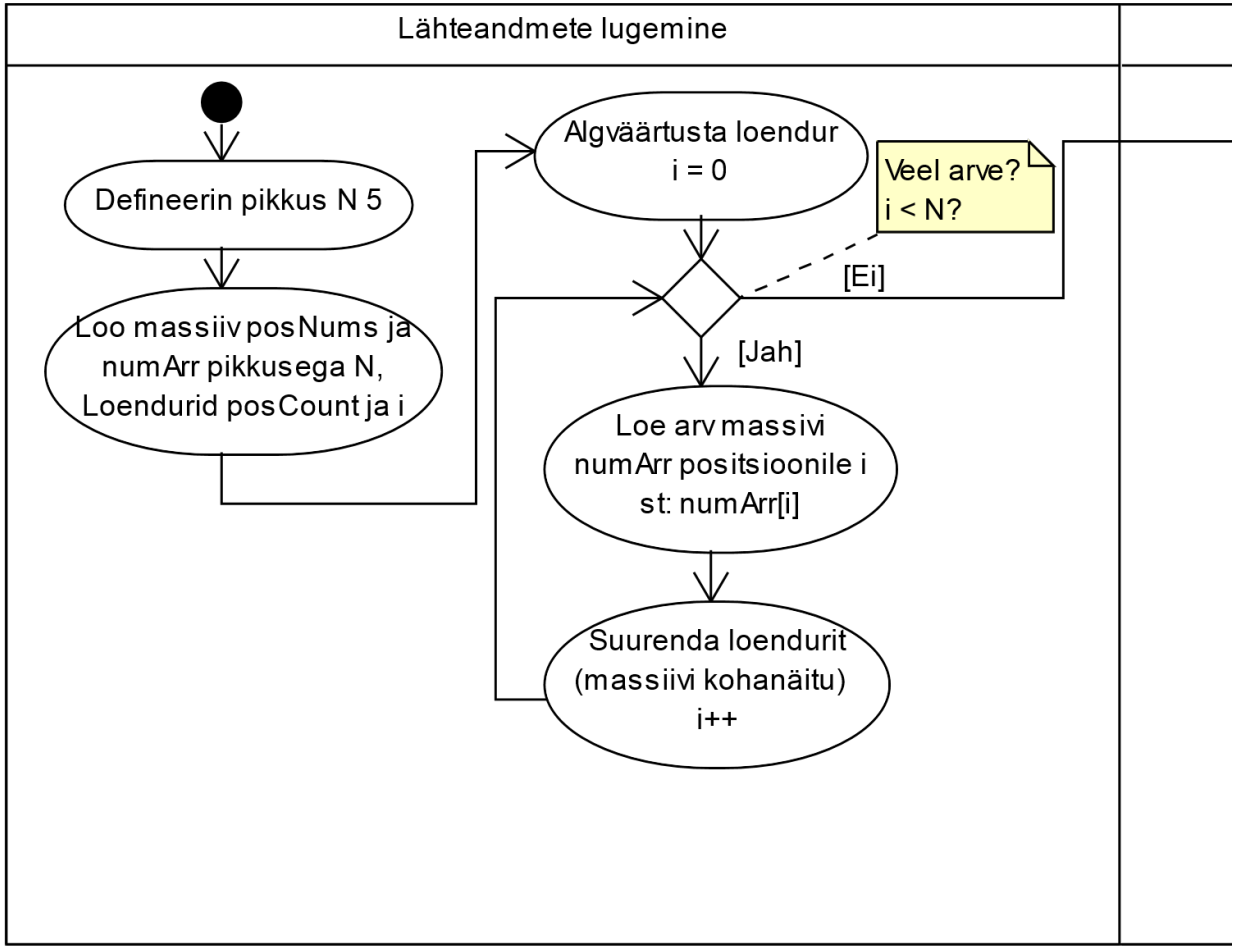
Klaviatuurilt loetakse sisse N täisarvu ning salvestatakse massiivi. Sealt tuleb välja noppida kõik positiivsed arvud ning need kopeerida teise, tühja, massiivi. Seejärel kuva tulemus.

Mõttekohad!?

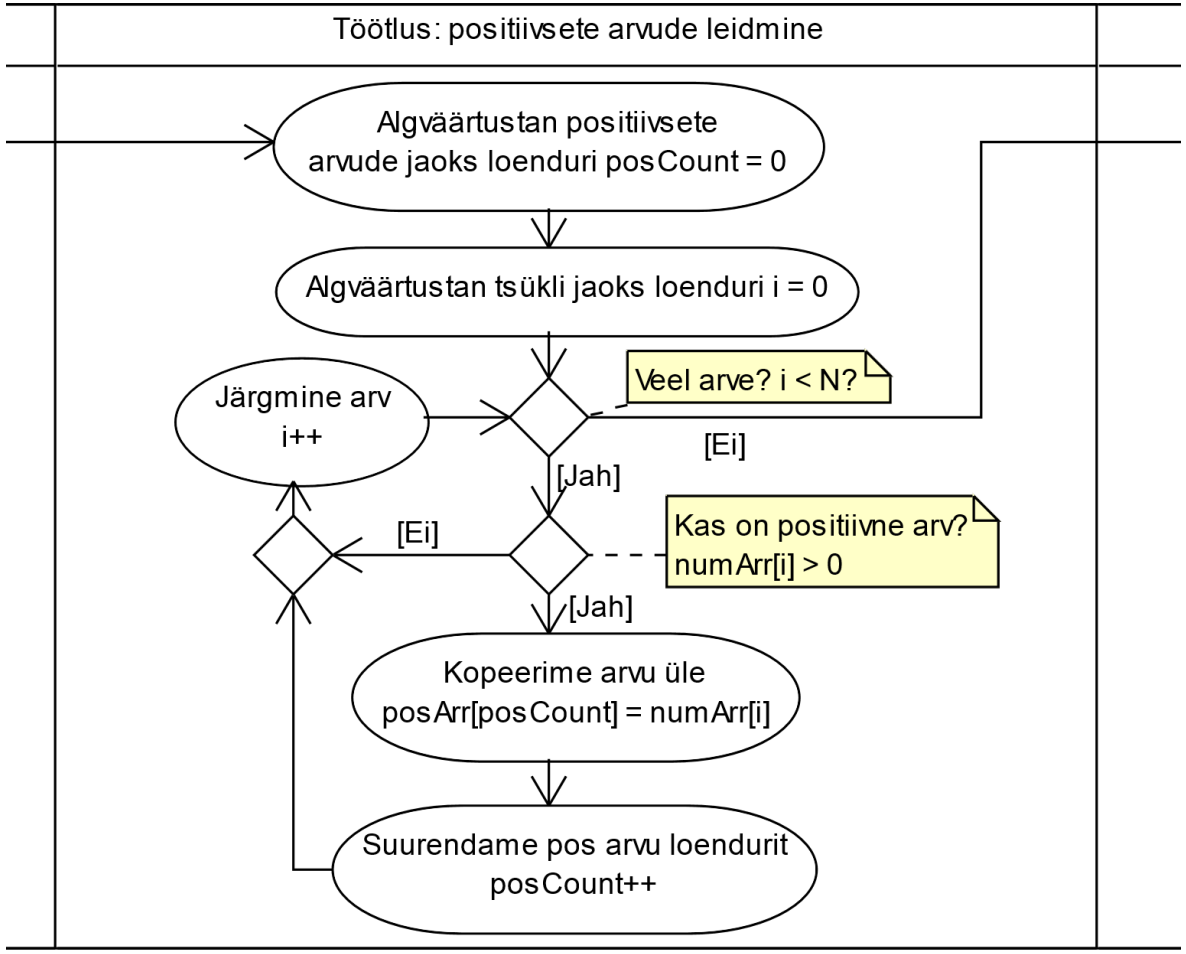
- Kust tuleb N?
- Mis tüüpi arvudega tegu on?
- Eksisteerib vajadus kahe massiivi järgi
- Kui pikk on teine massiiv?
- Kuidas indekseerida teist massiivi?
- Mis sammuga ja millal indeksid suurenevad?
- Mis saab nullidest?
- …



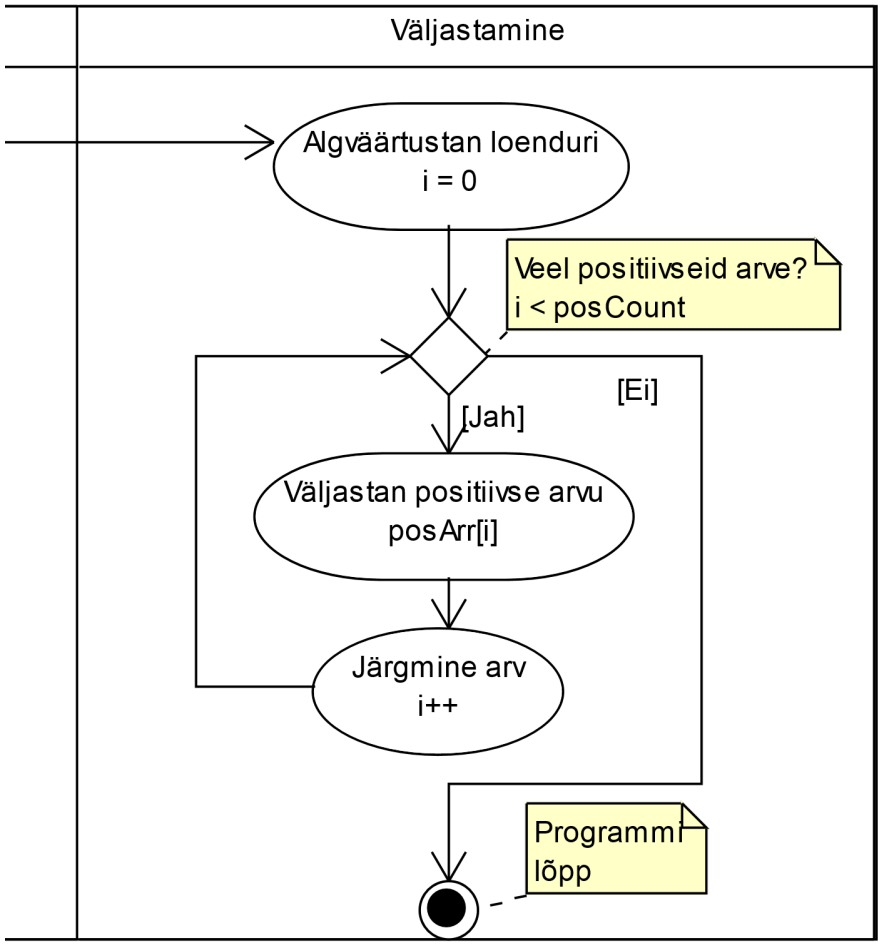
Lahendus UMLis (1/3)



Lahendus UMLis (2/3)



Lahendus UMLis (3/3)



Lahendus koodina

```c
#include <stdio.h>

#define NUM_COUNT 5

int main(void)
{
    int numArr[NUM_COUNT];
    int posNums[NUM_COUNT], i;
    int posCount = 0;

    for (i = 0; i < NUM_COUNT; i++)
    {
        scanf("%d", &numArr[i]);
    }
    for (i = 0; i < NUM_COUNT; i++)
    {
        if (numArr[i] > 0)
        {
            posNums[posCount] = numArr[i];
            posCount++;
        }
    }
    for (i = 0; i < posCount; i++)
    {
        printf("%d\n", posNums[i]);
    }
    return 0;
}
```

Kahe massiivi paralleelindekseerimine

```c
#include <stdio.h>

#define STUDENT_COUNT 4
#define NAME_LEN 7

int main(void)
{
    int grades[STUDENT_COUNT] = {5, 5, 3, 4};
    char names[STUDENT_COUNT][NAME_LEN] = {"Mari", "Alex", "Teele", "Kaarel"};
    int i;

    for (i = 0; i < STUDENT_COUNT; i++)
    {
        printf("Student %s got a grade of %d\n", names[i], grades[i]);
    }
    return 0;
}
```

Kahedimensiooniline massiiv (1)

Tuntud ka maatriksi nime all

Deklareerides peame määrama mõlema dimensiooni pikkused

Dimensioonid võivad olla erineva pikkusega (nt 3x10)

Indekseerimisel kasutame kahte indeksit

Üldjuhul on esimene reaindeks, teine veeruindeks, st

- `massiiv[rida][veerg]`

Kujuta ette – malelaud, Exceli tabel, kordinaatteljestik, …

Kahedimensiooniline massiiv (2)

Antud juhul on tegu **ruutmaatriksiga** (dimensioonide suurused võrdsed)

```
int massiiv[5][5]
```

massiiv[0][0]	massiiv[0][1]	massiiv[0][2]	massiiv[0][3]	massiiv[0][4]
massiiv[1][0]	massiiv[1][1]	massiiv[1][2]	massiiv[1][3]	massiiv[1][4]
massiiv[2][0]	massiiv[2][1]	massiiv[2][2]	massiiv[2][3]	massiiv[2][4]
massiiv[3][0]	massiiv[3][1]	massiiv[3][2]	massiiv[3][3]	massiiv[3][4]
massiiv[4][0]	massiiv[4][1]	massiiv[4][2]	massiiv[4][3]	massiiv[4][4]

2-dimensioonilise massiivi läbikäik Korrutustabeli loomine

```c
#include <stdio.h>

#define SUURUS 10

int main(void)
{
    int i, j, arvuMassiiv[SUURUS][SUURUS];

    for (i = 0; i < SUURUS; i++)
    {
        for (j = 0; j < SUURUS; j++)
        {
            arvuMassiiv[i][j] = (i + 1) * (j + 1);
        }
    }
    return 0;
}
```

Lisateemad (kui aega jääb)

Teoreetilised koos näidetega

- Arvusüsteemid ja nendevaheline teisendamine
- Kuidas saab koodist arvutile jooksutatav programm

Koodipraktika

- printf() kasutamise näiteid
- Andmetüüpide omadused, tüübiteisendused ja tehted