

# Функции

## (част 3)

Трифон Трифонов

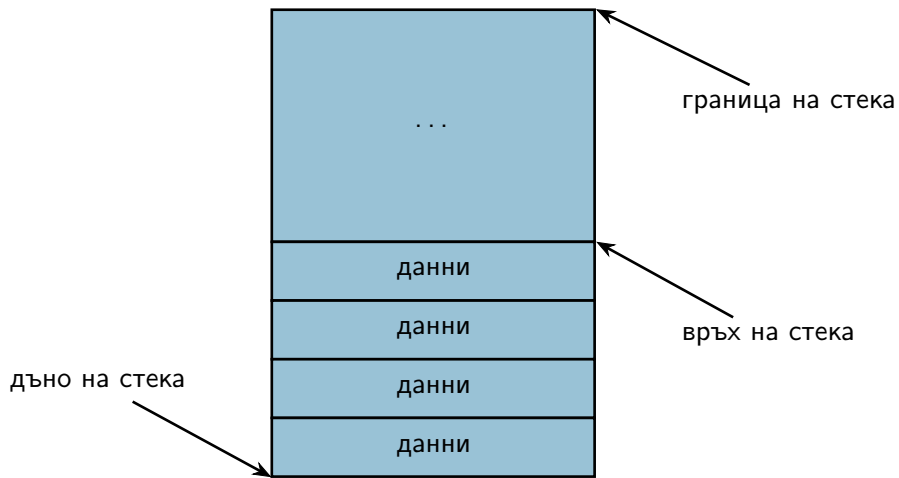
Увод в програмирането,  
спец. Компютърни науки, 1 поток, 2018/19 г.

20 декември 2018 г.

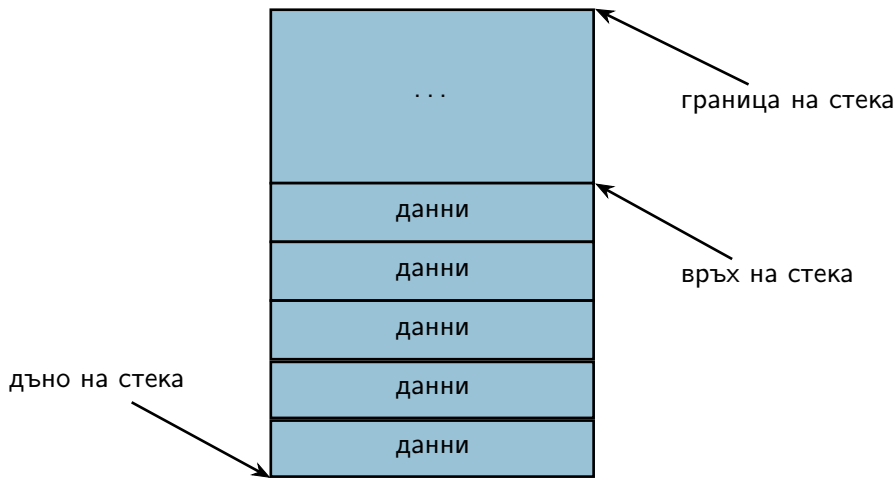
# Схема на програмната памет



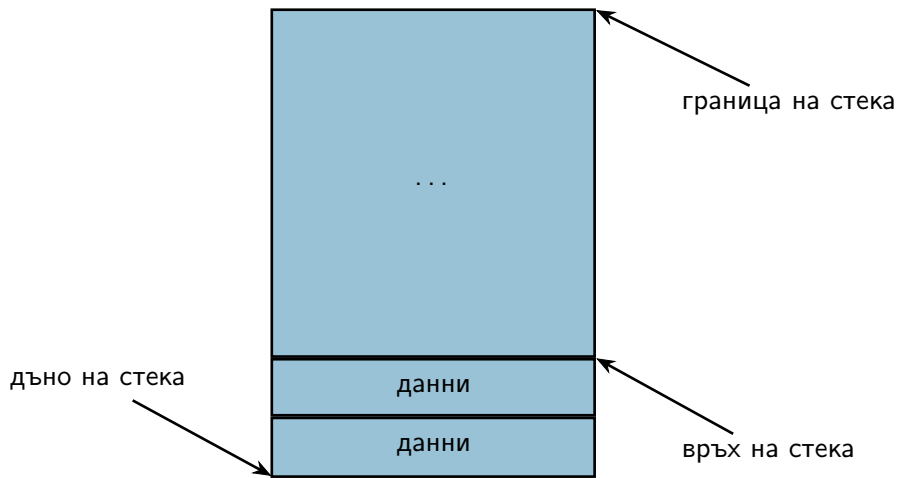
## Програмен стек



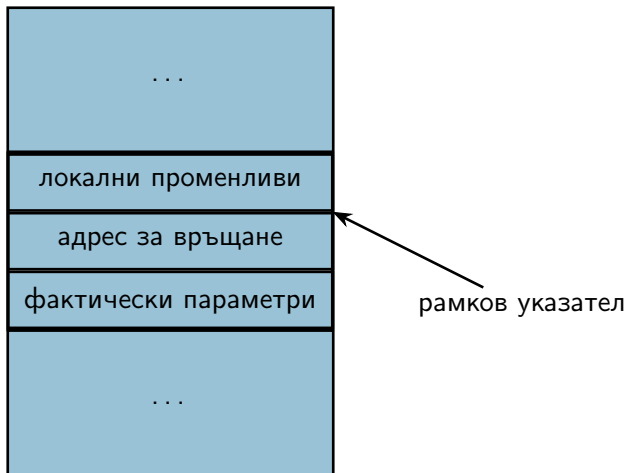
## Програмен стек



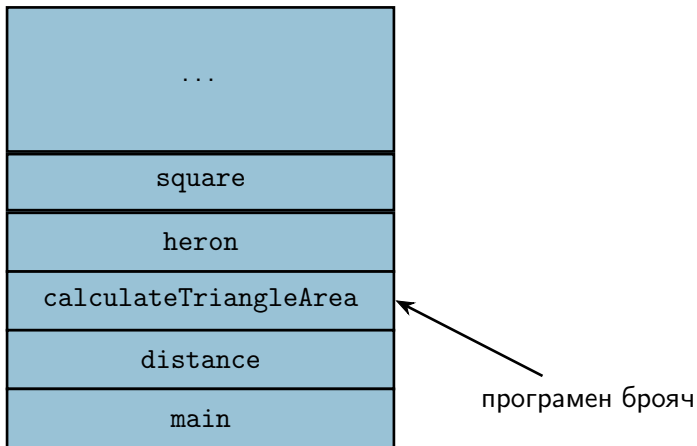
## Програмен стек



# Стекова рамка на функция



## Област за програмен код



# Предаване по стойност (call by value)

- пресмята се стойността на фактическия параметър



# Предаване по стойност (call by value)

- пресмята се стойността на фактическия параметър
- в стековата рамка на функцията се създава **копие** на стойността

# Предаване по стойност (call by value)

- пресмята се стойността на фактическия параметър
- в стековата рамка на функцията се създава **копие** на стойността
- всяка промяна на стойността остава локална за функцията

# Предаване по стойност (call by value)

- пресмята се стойността на фактическия параметър
- в стековата рамка на функцията се създава **копие** на стойността
- всяка промяна на стойността остава локална за функцията
- при завършване на функцията, предадената стойност и всички промени над нея **изчезват**

## Предаване с препратка (call by reference)

- Понякога искаме промените във **формалните** параметри да се отразят във **фактическите** параметри

## Предаване с препратка (call by reference)

- Понякога искаме промените във **формалните** параметри да се отразят във **фактическите** параметри
- Тогава трябва да обявим, че искаме фактическите параметри да могат да бъдат променявани

## Предаване с препратка (call by reference)

- Понякога искаме промените във **формалните** параметри да се отразят във **фактическите** параметри
- Тогава трябва да обявим, че искаме фактическите параметри да могат да бъдат променяни
- $\langle \text{параметър} \rangle ::= \langle \text{тип} \rangle \& \langle \text{идентификатор} \rangle$

## Предаване с препратка (call by reference)

- Понякога искаме промените във **формалните** параметри да се отразят във **фактическите** параметри
- Тогава трябва да обявим, че искаме фактическите параметри да могат да бъдат променяни
- $\langle \text{параметър} \rangle ::= \langle \text{тип} \rangle \& \langle \text{идентификатор} \rangle$
- **Примери:**

## Предаване с препратка (call by reference)

- Понякога искаме промените във **формалните** параметри да се отразят във **фактическите** параметри
- Тогава трябва да обявим, че искаме фактическите параметри да могат да бъдат променяни
- `<параметър> ::= <тип>& <идентификатор>`
- **Примери:**
  - `int add5(int& x) { x += 5; return x; }`



## Предаване с препратка (call by reference)

- Понякога искаме промените във **формалните** параметри да се отразят във **фактическите** параметри
- Тогава трябва да обявим, че искаме фактическите параметри да могат да бъдат променяни
- `<параметър> ::= <тип>& <идентификатор>`
- **Примери:**
  - `int add5(int& x) { x += 5; return x; }`
  - **фактическият параметър трябва да е lvalue!**

# Предаване с препратка (call by reference)

- Понякога искаме промените във **формалните** параметри да се отразят във **фактическите** параметри
- Тогава трябва да обявим, че искаме фактическите параметри да могат да бъдат променяни
- `<параметър> ::= <тип>& <идентификатор>`
- **Примери:**
  - `int add5(int& x) { x += 5; return x; }`
  - фактическият параметър трябва да е lvalue!
  - `add5(3);`

# Предаване с препратка (call by reference)

- Понякога искаме промените във **формалните** параметри да се отразят във **фактическите** параметри
- Тогава трябва да обявим, че искаме фактическите параметри да могат да бъдат променяни
- `<параметър> ::= <тип>& <идентификатор>`
- **Примери:**
  - `int add5(int& x) { x += 5; return x; }`
  - фактическият параметър трябва да е lvalue!
  - `add5(3);`
  - `int a = 3; cout << add5(a) << ' ' << a;`

*hence int &x = a;*

## Пример за предаване с препратка

Размяна на две променливи

```
void swap(int& x, int& y) {  
    int tmp = x;  
    x = y;  
    y = tmp;  
}
```

## Пример за предаване с препратка

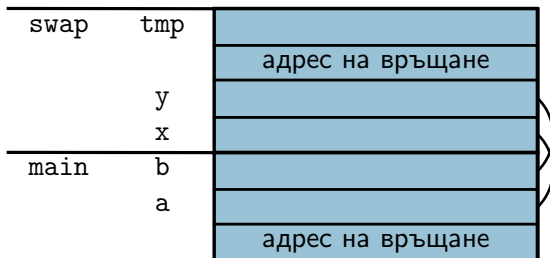
Размяна на две променливи

```
void swap(int& x, int& y) {  
    int tmp = x;  
    x = y;  
    y = tmp;  
}
```

*int &x = a, &y = b;*

```
int main() {  
    int a = 5, b = 8;  
    swap(a, b);  
    cout << a << ' ' << b << endl;  
}
```

## Стекова рамка при предаване с препратки



# Предаване по указател/адрес (call by pointer)

- Предава се **адрес** вместо стойност

## Предаване по указател/адрес (call by pointer)

- Предава се **адрес** вместо стойност
- Фактическите параметри трябва да са от тип “указател към нещо”



## Предаване по указател/адрес (call by pointer)

- Предава се **адрес** вместо стойност
- Фактическите параметри трябва да са от тип “указател към нещо”
- Функцията може да променя стойности на външни за функцията променливи **през подадените ѝ указатели**

## Предаване по указател/адрес (call by pointer)

- Предава се **адрес** вместо стойност
- Фактическите параметри трябва да са от тип “указател към нещо”
- Функцията може да променя стойности на външни за функцията променливи **през подадените ѝ указатели**
- **Примери:**

# Предаване по указател/адрес (call by pointer)

- Предава се **адрес** вместо стойност
- Фактическите параметри трябва да са от тип “указател към нещо”
- Функцията може да променя стойности на външни за функцията променливи **през подадените ѝ указатели**

- **Примери:**

- `int add5(int* px) { *px += 5; return *px; }`

*Handwritten red text: Const with an arrow pointing to the asterisk in the code above.*

# Предаване по указател/адрес (call by pointer)

- Предава се **адрес** вместо стойност
- Фактическите параметри трябва да са от тип “указател към нещо”
- Функцията може да променя стойности на външни за функцията променливи **през подадените ѝ указатели**
- **Примери:**
  - `int add5(int* px) { *px += 5; return *px; }`
  - ~~`add5(3); add5(&3);`~~

# Предаване по указател/адрес (call by pointer)

- Предава се **адрес** вместо стойност
- Фактическите параметри трябва да са от тип “указател към нещо”
- Функцията може да променя стойности на външни за функцията променливи **през подадените ѝ указатели**
- **Примери:**
  - `int add5(int* px) { *px += 5; return *px; }`
  - ~~`add5(3); add5(&3);`~~
  - `int a = 3; cout << add5(&a) << ' ' << a;`

## Пример за предаване по указател

Размяна на две променливи

```
void swap(int* p, int* q) {  
    int tmp = *p;  
    *p = *q;  
    *q = tmp;  
}
```

## Пример за предаване по указател

Размяна на две променливи

```
void swap(int* p, int* q) {
    int tmp = *p;
    *p = *q;
    *q = tmp;
}

int main() {
    int a = 5, b = 8;
    swap(&a, &b);
    cout << a << ' ' << b << endl;
}
```





# Предаване на масиви като параметри

- $\langle \text{параметър\_масив} \rangle ::= \langle \text{тип} \rangle \langle \text{име} \rangle [ [ \langle \text{константен\_израз} \rangle ] ] \mid$   
 $\langle \text{тип} \rangle * \langle \text{име} \rangle$

# Предаване на масиви като параметри

- $\langle \text{параметър\_масив} \rangle ::= \langle \text{тип} \rangle \langle \text{име} \rangle [ [ \langle \text{константен\_израз} \rangle ] ] \mid$   
 $\langle \text{тип} \rangle * \langle \text{име} \rangle$
- ВСЪЩНОСТ...

# Предаване на масиви като параметри

- $\langle \text{параметър\_масив} \rangle ::= \langle \text{тип} \rangle \langle \text{име} \rangle [ [ \langle \text{константен\_израз} \rangle ] ] \mid$   
 $\langle \text{тип} \rangle * \langle \text{име} \rangle$
- всъщност...
- ...масивите се предават **по указател!**

# Предаване на масиви като параметри

- $\langle \text{параметър\_масив} \rangle ::= \langle \text{тип} \rangle \langle \text{име} \rangle [ [ \langle \text{константен\_израз} \rangle ] ] \mid \langle \text{тип} \rangle * \langle \text{име} \rangle$
- всъщност...
- ...масивите се предават **по указател!**
- ...затова размерът на масива в скобите се игнорира!

# Предаване на масиви като параметри

- $\langle \text{параметър\_масив} \rangle ::= \langle \text{тип} \rangle \langle \text{име} \rangle [ [ \langle \text{константен\_израз} \rangle ] ] \mid \langle \text{тип} \rangle * \langle \text{име} \rangle$
- всъщност...
- ...масивите се предават **по указател!**
- ...затова размерът на масива в скобите се игнорира!
- ...затова промените в винаги се отразяват в оригинала!

# Предаване на многомерни масиви като параметри

- $\langle \text{параметър\_многомерен\_масив} \rangle ::=$   
     $\langle \text{тип} \rangle \langle \text{име} \rangle \llbracket \langle \text{константа} \rangle \rrbracket \{ \llbracket \langle \text{константа} \rangle \rrbracket \} |$   
     $\langle \text{тип} \rangle (*\langle \text{име} \rangle) \{ \llbracket \langle \text{константа} \rangle \rrbracket \}$

# Предаване на многомерни масиви като параметри

- $\langle \text{параметър\_многомерен\_масив} \rangle ::=$   
     $\langle \text{тип} \rangle \langle \text{име} \rangle \llbracket \langle \text{константа} \rangle \rrbracket \{ \llbracket \langle \text{константа} \rangle \rrbracket \} |$   
     $\langle \text{тип} \rangle (*\langle \text{име} \rangle) \{ \llbracket \langle \text{константа} \rangle \rrbracket \}$
- многомерните масиви също се предават по указател

# Предаване на многомерни масиви като параметри

- $\langle \text{параметър\_многомерен\_масив} \rangle ::=$   
     $\langle \text{тип} \rangle \langle \text{име} \rangle \llbracket \langle \text{константа} \rangle \rrbracket \{ \llbracket \langle \text{константа} \rangle \rrbracket \} |$   
     $\langle \text{тип} \rangle (*\langle \text{име} \rangle) \{ \llbracket \langle \text{константа} \rangle \rrbracket \}$
- многомерните масиви също се предават по указател
- първата размерност се игнорира



# Предаване на многомерни масиви като параметри

- $\langle \text{параметър\_многомерен\_масив} \rangle ::=$   
 $\langle \text{тип} \rangle \langle \text{име} \rangle \llbracket \langle \text{константа} \rangle \rrbracket \{ \llbracket \langle \text{константа} \rangle \rrbracket \} |$   
 $\langle \text{тип} \rangle (*\langle \text{име} \rangle) \{ \llbracket \langle \text{константа} \rangle \rrbracket \}$
- многомерните масиви също се предават по указател
- първата размерност се игнорира
  - останалите трябва да се укажат, за да работи правилно указателната аритметика

# Предаване на многомерни масиви като параметри

- $\langle \text{параметър\_многомерен\_масив} \rangle ::=$   
     $\langle \text{тип} \rangle \langle \text{име} \rangle \llbracket \langle \text{константа} \rangle \rrbracket \{ \llbracket \langle \text{константа} \rangle \rrbracket \} |$   
     $\langle \text{тип} \rangle (*\langle \text{име} \rangle) \{ \llbracket \langle \text{константа} \rangle \rrbracket \}$
- многомерните масиви също се предават по указател
- първата размерност се игнорира
  - останалите трябва да се укажат, за да работи правилно указателната аритметика
- (поне) първата размерност трябва да се подава като параметър

# Предаване на многомерни масиви като параметри

- `<параметър_многомерен_масив> ::=`  
`<тип> <име> [[<константа>]] { [[<константа>]] } |`  
`<тип> (*<име>) { [[<константа>]] }`
- многомерните масиви също се предават по указател
- първата размерност се игнорира
  - останалите трябва да се укажат, за да работи правилно указателната аритметика
- (поне) първата размерност трябва да се подава като параметър
- **Внимание:** `int* a[10]` е различно от `int (*a)[10]`!

# Предаване на многомерни масиви като параметри

- $\langle \text{параметър\_многомерен\_масив} \rangle ::=$   
 $\langle \text{тип} \rangle \langle \text{име} \rangle \llbracket \langle \text{константа} \rangle \rrbracket \{ \llbracket \langle \text{константа} \rangle \rrbracket \} |$   
 $\langle \text{тип} \rangle (*\langle \text{име} \rangle) \{ \llbracket \langle \text{константа} \rangle \rrbracket \}$
- многомерните масиви също се предават по указател
- първата размерност се игнорира
  - останалите трябва да се укажат, за да работи правилно указателната аритметика
- (поне) първата размерност трябва да се подава като параметър
- **Внимание:** `int* a[10]` е различно от `int (*a)[10]`!
  - `int* a[10]`  $\iff$  масив от 10 указателя към цели числа



# Предаване на многомерни масиви като параметри

- `<параметър_многомерен_масив> ::=`  
`<тип> <име> [[<константа>]] { [<константа>] } |`  
`<тип> (*<име>) { [<константа>] }`
- многомерните масиви също се предават по указател
- първата размерност се игнорира
  - останалите трябва да се укажат, за да работи правилно указателната аритметика
- (поне) първата размерност трябва да се подава като параметър
- **Внимание:** `int* a[10]` е различно от `int (*a)[10]`!
  - `int* a[10]`  $\iff$  масив от 10 указателя към цели числа
  - `int (*a)[10]`  $\iff$  указател към масив от десет цели числа



# Предаване на многомерни масиви като параметри

- $\langle \text{параметър\_многомерен\_масив} \rangle ::=$   
 $\langle \text{тип} \rangle \langle \text{име} \rangle \llbracket \langle \text{константа} \rangle \rrbracket \{ \llbracket \langle \text{константа} \rangle \rrbracket \} |$   
 $\langle \text{тип} \rangle (*\langle \text{име} \rangle) \{ \llbracket \langle \text{константа} \rangle \rrbracket \}$
- многомерните масиви също се предават по указател
- първата размерност се игнорира
  - останалите трябва да се укажат, за да работи правилно указателната аритметика
- (поне) първата размерност трябва да се подава като параметър
- **Внимание:** `int* a[10]` е различно от `int (*a)[10]`!
  - `int* a[10]`  $\iff$  масив от 10 указателя към цели числа
  - `int (*a)[10]`  $\iff$  указател към масив от десет цели числа
  - ...но понеже масивите от тип T могат да се разглеждат като указатели към тип T...

# Предаване на многомерни масиви като параметри

- $\langle \text{параметър\_многомерен\_масив} \rangle ::=$   
 $\langle \text{тип} \rangle \langle \text{име} \rangle \llbracket \langle \text{константа} \rangle \rrbracket \{ \llbracket \langle \text{константа} \rangle \rrbracket \} |$   
 $\langle \text{тип} \rangle (*\langle \text{име} \rangle) \{ \llbracket \langle \text{константа} \rangle \rrbracket \}$
- многомерните масиви също се предават по указател
- първата размерност се игнорира
  - останалите трябва да се укажат, за да работи правилно указателната аритметика
- (поне) първата размерност трябва да се подава като параметър
- **Внимание:** `int* a[10]` е различно от `int (*a)[10]`!
  - `int* a[10]`  $\iff$  масив от 10 указателя към цели числа
  - `int (*a)[10]`  $\iff$  указател към масив от десет цели числа
  - ...но понеже масивите от тип T могат да се разглеждат като указатели към тип T...
  - `int (*a)[10]`  $\iff$  масив от масив от десет цели числа

# Предаване на многомерни масиви като параметри

- $\langle \text{параметър\_многомерен\_масив} \rangle ::=$   
 $\langle \text{тип} \rangle \langle \text{име} \rangle \llbracket \langle \text{константа} \rangle \rrbracket \{ \llbracket \langle \text{константа} \rangle \rrbracket \} |$   
 $\langle \text{тип} \rangle (*\langle \text{име} \rangle) \{ \llbracket \langle \text{константа} \rangle \rrbracket \}$
- многомерните масиви също се предават по указател
- първата размерност се игнорира
  - останалите трябва да се укажат, за да работи правилно указателната аритметика
- (поне) първата размерност трябва да се подава като параметър
- **Внимание:**  $\text{int}^* \text{ a}[10]$  е различно от  $\text{int} (*\text{a})[10]!$ 
  - $\text{int}^* \text{ a}[10] \iff$  масив от 10 указателя към цели числа
  - $\text{int} (*\text{a})[10] \iff$  указател към масив от десет цели числа
  - ...но понеже масивите от тип T могат да се разглеждат като указатели към тип T...
  - $\text{int} (*\text{a})[10] \iff$  масив от масив от десет цели числа
  - $\text{int} (*\text{a})[10] \iff$  двумерен масив от цели числа с 10 колони

*↖ [ ] [ ] ↗*

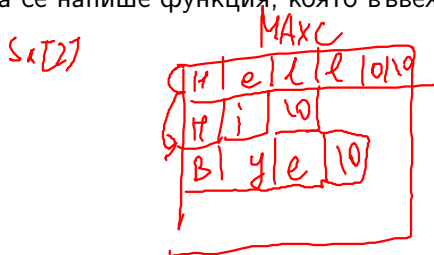


# Примерни функции

- 1 Да се напише функция, която извежда матрица от числа

# Примерни функции

- 1 Да се напише функция, която извежда матрица от числа
- 2 Да се напише функция, която въвежда масив от низове



# Примерни функции

- 1 Да се напише функция, която извежда матрица от числа
- 2 Да се напише функция, която въвежда масив от низове
- 3 Да се напише функция, която проверява дали дадена дума се съдържа в масив от низове

# Примерни функции

$$c_{ij} = \vec{a}_i \cdot \vec{b}_j = \sum_{\ell=0}^{n-1} a_{i\ell} \cdot b_{\ell j}$$

- 1 Да се напише функция, която извежда матрица от числа
- 2 Да се напише функция, която въвежда масив от низове
- 3 Да се напише функция, която проверява дали дадена дума се съдържа в масив от низове
- 4 Да се напише функция, която умножава две правоъгълни матрици

$$\begin{matrix} m & & n & & & & k \\ & & \left( \begin{array}{c} \\ \\ \\ \end{array} \right) & & \left( \begin{array}{c} \\ \\ \\ \end{array} \right) & & \\ & & A & & B & & \\ & & & & = & & m \left( \begin{array}{c} \\ \\ \\ \end{array} \right) \\ & & & & & & C \end{matrix}$$