Programming languages

Functions
Arrays and pointers
Strings

Functions and procedures

- In a sense, everything is already known and clear:

```
 7  double pot(double podstawa, int wykladnik)
 8  {
 9      double wynik= 1;
10      for(int i = 0; i < wykladnik; ++i) wynik=podstawa*wynik;
11      return wynik;
12  }
13
14
15  int main(int argc, char *argv[])
16  {
17  cout << "2 do potegi 3 wynosi " << pot(2,3)<<endl;
18  system("PAUSE");
19  return 0;
```

Functions and procedures

- In a sense, because:
  1. You can return several values at once (**reference**)☺
  2. Parameters can be passed by **reference** or by **value**
  3. Functions can have a **variable** number of parameters.
  4. We have **inline** functions at our disposal

Reference vs value

```
 7   void wartosc(int co)
 8   {
 9       ++co;
10      return;
11   }
12
13   void referencja(int &co)
14   {
15       ++co;
16      return;
17   }
18
```

```
int main(int argc, char *argv[])
{
    int liczba = 0;
cout << "Na poczatku liczba wynosi: "<< liczba << endl;
wartosc(liczba);
cout << "Ale wywolujemy funkcje wartosc i liczba wynosi juz: " << liczba << endl;
referencja(liczba);
cout << "Lecz dopiero referencja zmieni wartosc na " << liczba <<endl;
system("PAUSE");
return 0;
}
```

Exercise – functions

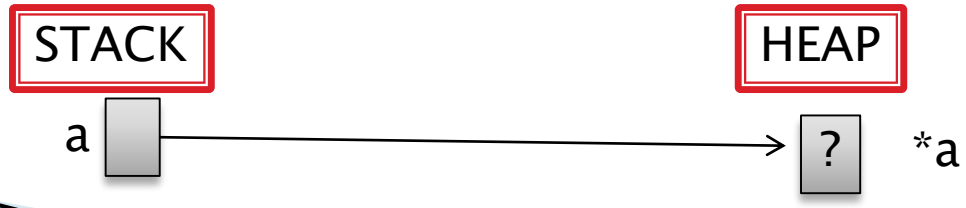
1. Design a function that reads numbers from the user until the value 0 is given. The function should return 0 or 1 – depending on whether the number amount is an even or an odd number. What's more – the function has to return the sum and the product of all given numbers at the same time.
2. Write a function that returns the value of the n-th element of the Fibonacci sequence.

Stack vs heap

- The advantages of using dynamic memory allocation:
  - It can be shared between different objects in the program.
  - Its size can be determined while the program is running.

```
int a = 2;
```


```
int* a = new int;
```



Pointers in C++

- Pointer variable – a variable that stores the address of another programming entity (object, variable) in the memory.
- The pointer must have a given type it points to.

```
1  int jakasLiczba = 10;
2  int* jakisWskaznik;
3
4  jakisWskaznik = &jakasLiczba
5  std::cout << *jakisWskaznik;
```

Pointers in C++

```
1  int jakasLiczba = 10;
2  int* jakisWskaznik;
3
4  jakisWskaznik = &jakasLiczba
5  std::cout << *jakisWskaznik;
```

Ln1: declaration of an integer variable
Ln2: declaration of a pointer to an integer
Ln4: assigning a variable address to the pointer
Ln5: dereference of the value indicated by the pointer

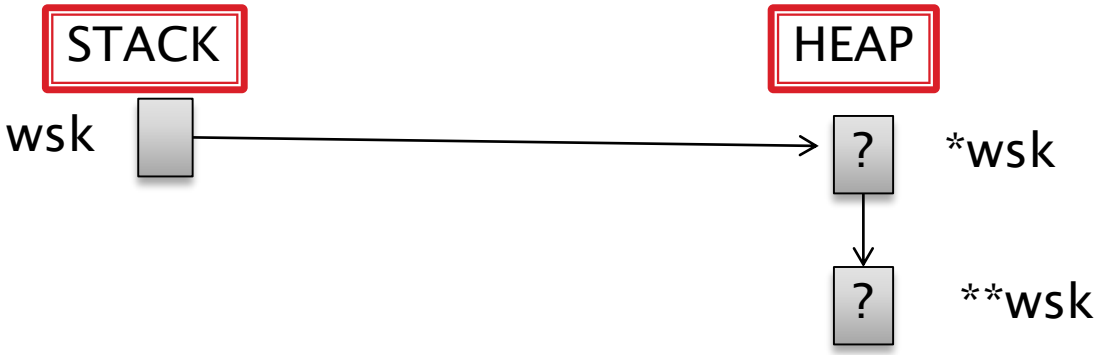
Pointers in C++

```
9  int jakasLiczba = 10;
10 int* jakisWskaznik;
11
12 jakisWskaznik = &jakasLiczba;
13 cout << "Adres wskaznika: " << jakisWskaznik << endl;
14 cout << "Adres zmiennej: " << &jakasLiczba <<endl;
15 cout << "Wartosc: " << *jakisWskaznik << endl;
16
```

```
Adres wskaznika: 0x22ff44
Adres zmiennej: 0x22ff44
Wartosc: 10
Aby kontynuować, naciśnij
```

Allocation and deallocation

```
int** wsk;
wsk = new int*;
*wsk = new int;
```




```
new int; // Ops... I leaked it again...
```


```
int* liczba = new int;
delete liczba;
```

There should be one **delete** for each **new**. **ALWAYS**.

Arrays – plain types

- Similar to Javy:

type name[size]

These types of arrays are created on the stack.

```
21    typ nazwa_tablicy[rozmiar];
22
23    int tablica[10];
24
25    int tablica[4] = {2,4,6,8};
26
27    int tablica[100] = {1,};
28
29    int tablica[] = {1, 2, 3, 4, 5, 6, 7, 10};
30
```

```
int tablica[] = { [3] = 10, 20, 30, [4] = 50};
```

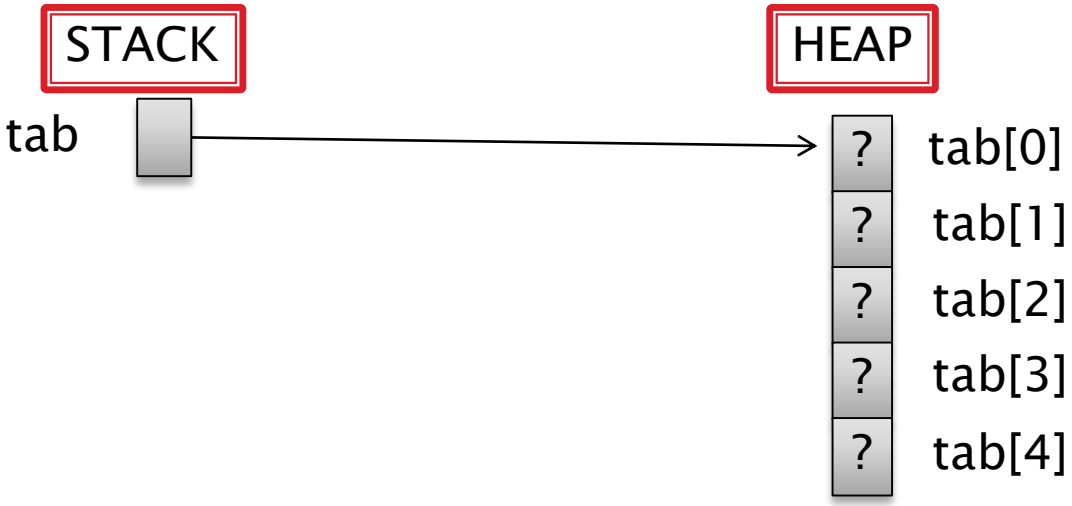
```
23  int main(int argc, char *argv[])
24  {
25      int tablica[5] = {1,2,3,4,5};
26      for (int i=0; i<(sizeof tablica / sizeof *tablica); ++i) cout << tablica[i] <<endl;
27      system("PAUSE");
28      return 0;
29  }
```

Passing arrays to a function

```
 7 void test(int tab[]) {
 8       for(int i=0;i<10;++i) tab[i]+=1;
 9       }
10
11 int main(int argc, char *argv[])
12 {
13 int tab[10]={1,1,1,1,1,1,1,1,1,1};
14 cout << tab[0] << endl;
15 test(tab);
16 cout << tab[0] << endl;
17
18 system("PAUSE");
19 return 0;
```

Dynamically allocated arrays

```
int* tab = new int[5];
delete[] tab;
```

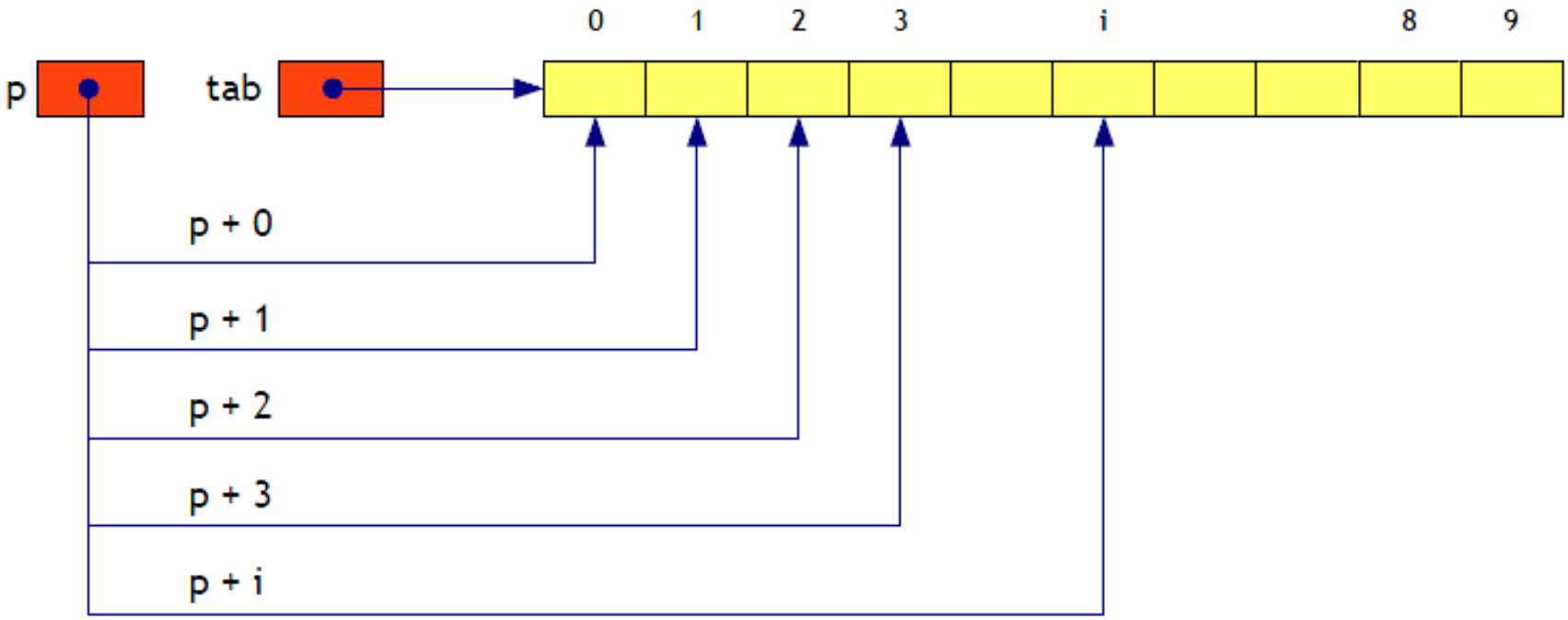


The name of the array and the indicator to its beginning

```
tab[ 0 ] = 5;
tab[ 1 ] = 1;
tab[ 2 ] = 10;
. . .
tab[ i ] = 22;
```

Equivalent

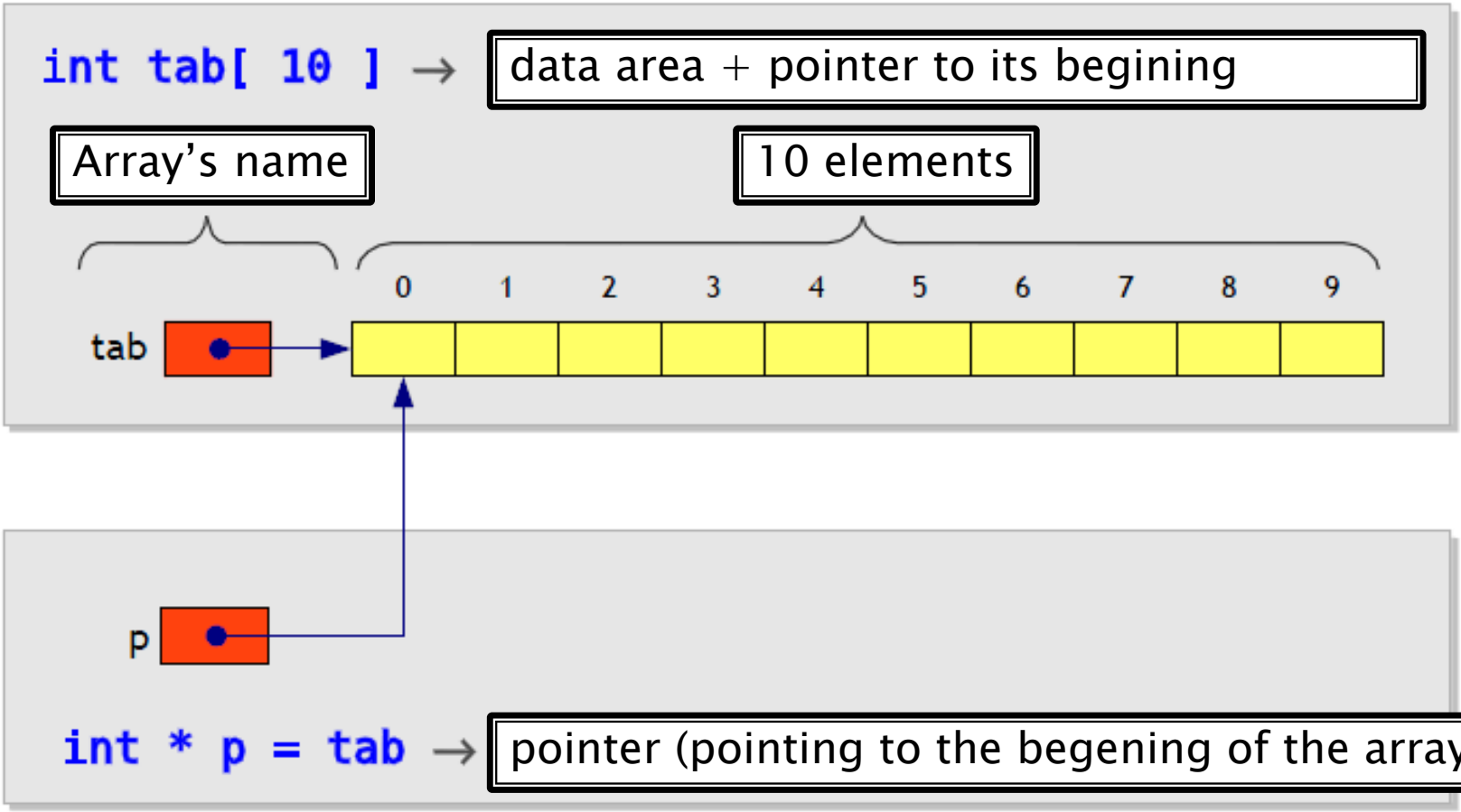
```
*p = 5
*( p + 1 ) = 1
*( p + 2 ) = 10
. . .
*( p + i ) = 22
```



The pointer is not an array!

```
int tab[ 10 ];
int * p = tab;
```

This is not the same!



Array name

You are not allowed to modify the array name

```
int tab[ 10 ];
int * p = tab;


tab = p;
tab++;


p = tab + 8;
p++;
```

☒ Źle

☑ OK

Pointers and arrays

- In C ++, you can use pointer arithmetic (add or subtract integers to its value). The compiler itself multiplies the number enlarged by the size of the pointer type to add the appropriate number of bytes to the address.

```
1  int tab[4];      //tab jest typu int *
2  char tab2[4][7]; //tab2 jest typu char **
3  //jest to podwójny wskaźnik bo tablica jest dwuwymiarowa
4
5  *tab = 0;          //równoznaczne z tab[0]=0
6  *(tab+1) = 5;      //równoznaczne z tab[1]=5
7  **tab2 = 'a';         //równoznaczne z tab2[0][0]='a'
8  *(tab2[4]+5) = 'c';  //równoznaczne z tab2[4][5]='c'
```

Pointers and arrays

- Referring to a one-dimensional array:

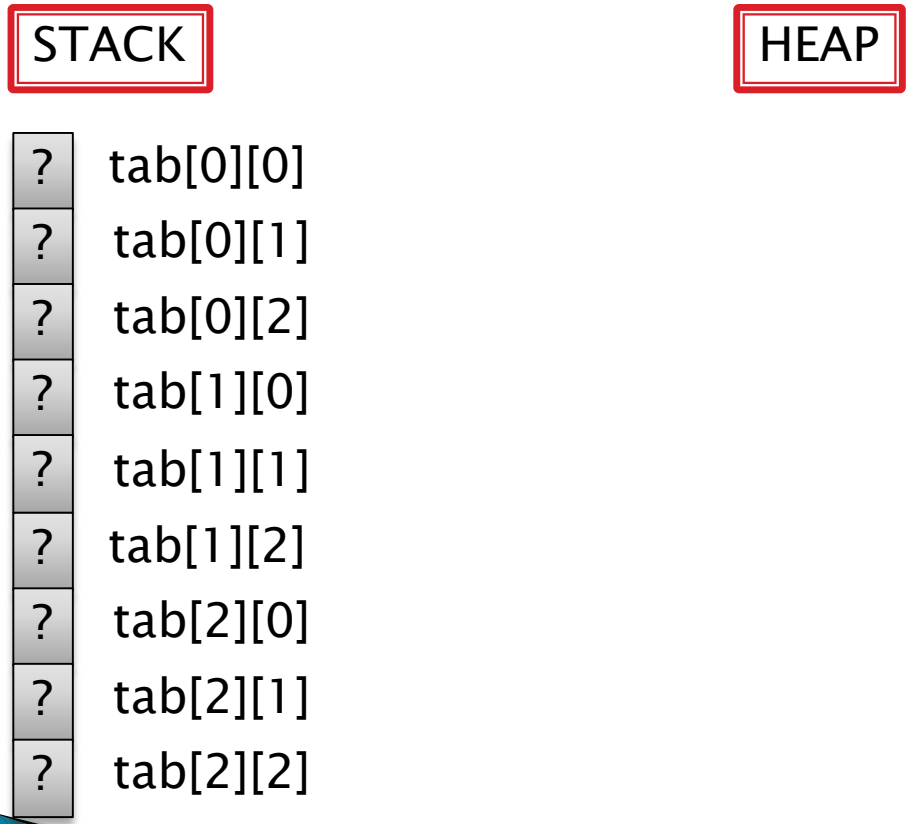
```
int tab[6];
int* wskTab = tab;
wskTab[4] = 1;
```

- Passing arrays to functions:

```
void Przetwarzaj(int* tablica) { // (int tablica[])
  for (int i = 0; i < 5; i++) {
      tablica[i] = -5;
  }
}
int* tab_dyn = new int[2];
tab[0] = 2;
tab[1] = 3;
int tab_stos[] = {2, 3};
Przetwarzaj(tab_dyn);
Przetwarzaj(tab_stos);
Przetwarzaj(&tab_stos[0]);
```

Multidimensional arrays

```
char statki[3][3];
statki[1][1] = 'X';
```

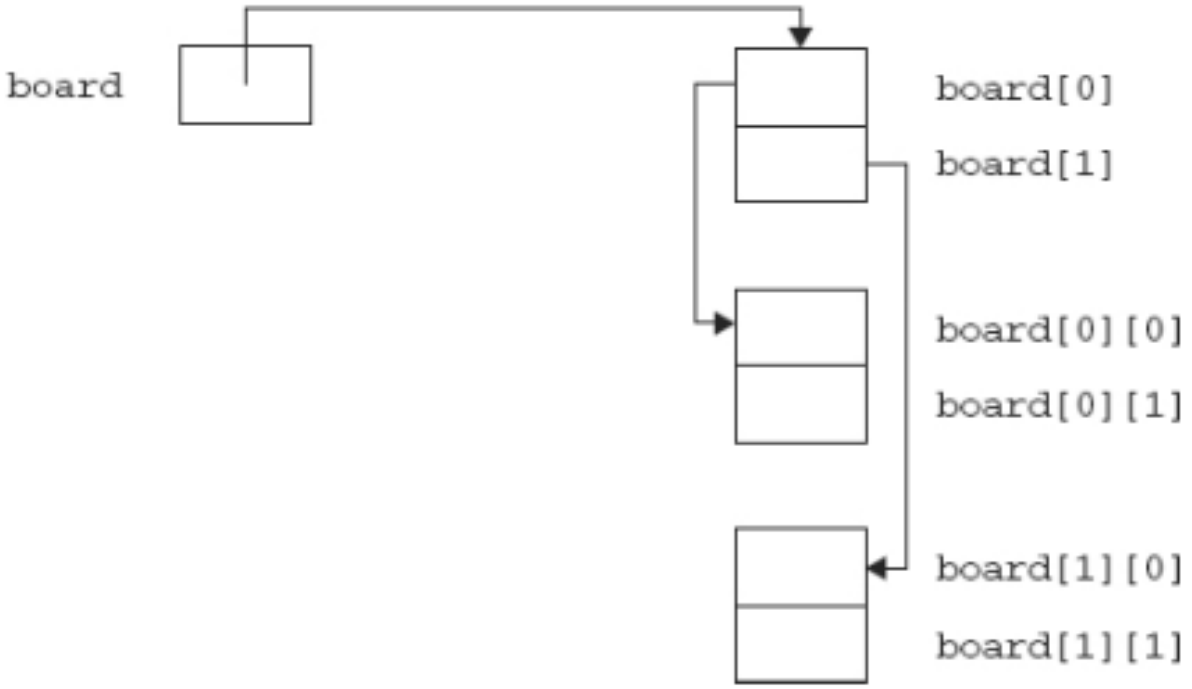


Multidimensional array allocated dynamically

- Is it possible?

```
const int i = 2, j = 2;
char** board = new char[i][j];
```

STACK HEAP



Exercises

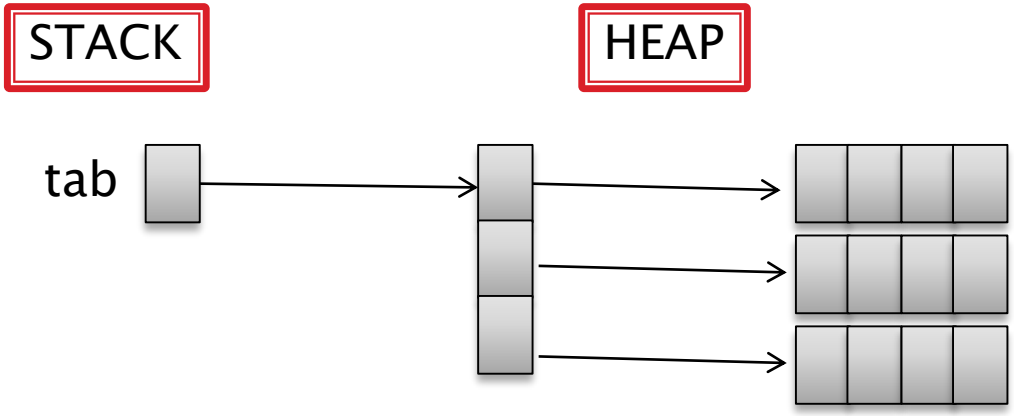
1. Declare an 100-element array. Fill the table according to the rules of the Fibbonacci sequence.
2. Write a program that converts the decimal number given by the user to a binary and hexadecimal number.
3. Write a program to multiply two matrices of any size. Find out if multiplication is possible!
4. Write a „guess my number” game. The program draws a number from the 1 ... 100 range, and our task is to guess the number based on "too much", "too little". After guessing, the program displays the number of attempts.
5. Write the entire ASCII table to the screen (each character in the new line with a number) and save it to the table named tab_ASCII in the program.

Use a dynamically allocated array in all execrices.

Dynamically allocated two-dimensional array

- Correct design (3 x 4)

```
char** tab = new char*[3];
for (size_t i = 0; i < 3; i++)
  tab[i] = new char[4];
```




```
for (size_t i = 0; i < 3; i++)
  delete[] tab[i];
delete[] tab;
```

Exercises

1. Write a game of ships on a dynamically generated board (up to 10 x 10). The game is for one user.

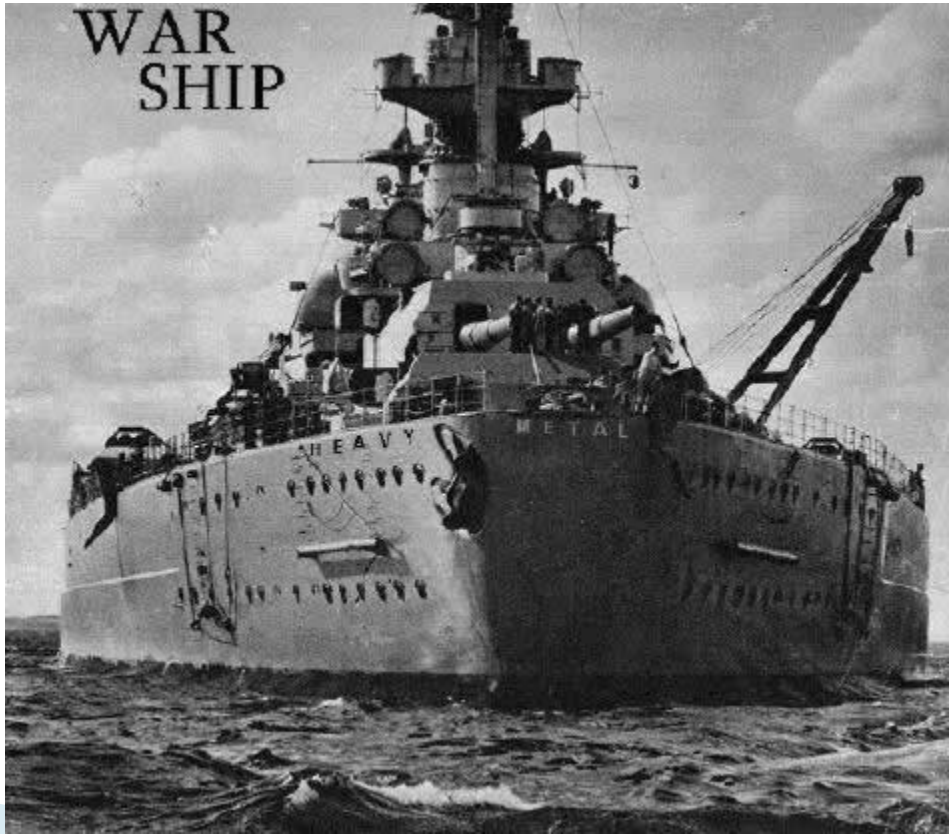
   Allowed ships:

   4 x one-mast

   Allowed situations:

   „Hit – sink”,

   „Miss”,



Interesting facts

```
1   #include <iostream>
2
3   using namespace std;
4
5   void suma(int tab[200][2]) {
6       tab[200][1] = 5;
7       cout << tab[0][1];
8   }
9
10  int main()
11  {
12      int tab[2][2] = {{1,2},{3,4}};
13      suma(tab);
14      return 0;
15  }
16
```

```
int main()
{
    int tab[10];
    int * wsk;
    wsk = tab + 1 ;
    return 0;
}
```

```
int main()
{
    int tab[ 10 ];
    tab = tab + 1;
    return 0;
}
```

Multidimensional arrays

```
1   #include <iostream>
2
3   using namespace std;
4
5   void suma(int tab[][2][2]) {
6       tab[1][1][1] = 5;
7       cout << tab[1][1][1] << endl;
8   }
9
10  int main()
11  {
12      int tab[2][2][2] = {{{1,2},{3,4}},{{5,6},{7,8}}};
13      suma(tab);
14      return 0;
15  }
16
```

0-terminated strings

- Strings treated as arrays ending with a character with code 0:

```
1   #include <cstdlib>
2   #include <iostream>
3   #include <cstring>
4
5   using namespace std;
6
7   int main(int argc, char** argv) {
8
9   char tekst[90] = "Ten tekst musi byc krotszy niz 89 znakow";
10  char nap[10] = "123456789";
11  char napis[] = "Ala ma kota";
12
13  cout << tekst << endl << napis <<endl << nap << endl;
14
```

Important note

A string literal can be assigned to an array only at its first initialization.

```
char napis[ N ] = "C/C++";  // OK
. . .
napis = "C/C++";   // Błąd, niedozwolone przypisanie
napis = "";        // Błąd, niedozwolone przypisanie
```

You are also not allowed to:

```
char napis[ N ] = "C/C++";  // OK
char napis1[ N ] = napis;   // Błąd, niedozwolona inicjalizacja
```

0-terminated strings

- Compare: **strcmp**(str1, str2): returns -, + or 0.
- Copy: **strcpy**(str1, str2) – something like str1 = str2.
- Concatenation: **strcat**(str1, str2) – something like: str1 = str1 + str2.
- Character count: **strlen**(str)
- Conversions:
  - **atol**, **strtol** – string to long
  - **atoi** – string to integer
  - **atoll**, **strtoll** – string to long long
  - **atof**, **strtod** – string to float/double

http://en.cppreference.com/w/cpp/string/byte

What's wrong?

```
char* kopiujStr(const char* wejscie) {
  char* wynik = new char[strlen(wejscie)];
  strcpy(wynik, wejscie);
  return wynik;
}
```

- Correct version:

```
char* kopiujStr(const char* wejscie) {
  char* wynik = new char[strlen(wejscie) + 1];
  strcpy(wynik, wejscie);
  return wynik;
}
```

Interesting cases

- What will be assigned?

```
char text1[] = "abcdef";
size_t s1 = sizeof(text1);
size_t s2 = strlen(text1);
char* text2 = "abcdef";
size_t s3 = sizeof(text2);
size_t s4 = strlen(text2);
```

Data valitation

```
#include <cctype>
#include <clocale>
```

Function name	Description
isalnum	Checks if the character is alphanumeric
isalpha	Checks if the character is a valid letter
islower	Checks if the character is a lowercase letter
isupper	Checks if the character is a capital letter
isdigit	Checks if the character is a number
isxdigit	Checks if the character is a hexadecimal number
iscntrl	Checks if the character is a control character
isgraph	Checks if the character has a graphical representation
isspace	Checks if the character is a white character
isblank *(C++11)*	Checks if the character is a white space separating words
isprint	Checks whether the character can be displayed
ispunct	Checks whether the character is a stop sign

Function name	Description
tolower	Returns the character as a lowercase letter
toupper	Returns the character as a capital letter

Excercise

1. Using a 20-element character array, create a program that converts lowercase letters to uppercase and vice versa in the text given by the user and display the text after the changes on the screen.

Strings: a better way

- In C ++, there is a standard string type!

```
9  string s;
10 cin >> s; //sproboj tu podac cos ze spacja :)
11 cin.ignore();
12 fflush(stdin);
13 cout << s << endl;
14 getline(cin,s);
15 cout << s << endl;
16 string drugi="Ala ma kota";
17 if(s==drugi) cout << "Yupi!"; else cout << "Niestety";
18 cout << endl;
19 s+=" i psa";
20 cout << s << endl;
```

```
35
36 string zdanie="Byc, albo nie byc - oto jest pytanie";
37 cout << zdanie <<endl;
38 for(int i=0;i<zdanie.size();i+=2) { //pokemonizator
39         zdanie.at(i) = toupper(zdanie.at(i));
40         }
41 cout << zdanie;
```

```
24 string csv;
25 do {
26      cin >> csv;
27      cout << csv << endl;
28 }
29 while(csv!=";");
30
31 csv="";
32 getline(cin,csv,'$');
33 cout << csv << endl;
```

String conversion

- Convert to string:

```
string to_string(int val);
string to_string(unsigned val);
string to_string(long val);
string to_string(unsigned long val);
string to_string(long long val);
string to_string(unsigned long long val);
string to_string(float val);
string to_string(double val);
string to_string(long double val);
```

- From string to …

```
int stoi(const string& str, size_t *idx=0, int base=10);
long stol(const string& str, size_t *idx=0, int base=10);
unsigned long stoul(const string& str, size_t *idx=0, int base=10);
long long stoll(const string& str, size_t *idx=0, int base=10);
unsigned long long stoull(const string& str, size_t *idx=0, int
  base=10);
float stof(const string& str, size_t *idx=0);
double stod(const string& str, size_t *idx=0);
long double stold(const string& str, size_t *idx=0);
```

Strings: a better way

- Other methods:
  - str1.**compare(str)** – works the same as the strcmp
  - str1.**c_str()** – returns a 0-terminated string
  - str1.**find(what, [start_from])**, rfind (last occurence),
  - find_first_not_of, find_first_of, …
  - str1.**replace(start,length,sourceForReplacement)**
  - str1.**substr(start_from, length)** – returns a substring
  - str1.**swap(str)** – swaps the contents of two strings

http://en.cppreference.com/w/cpp/string/basic_string

Excercises

1. Implement simple scout ciphers: GA-DE-RY-PO-LU-KI, PO-LI-TY-KA-RE-NU, KA-CE-MI-NU-TO-WY. A menu is displayed where the user selects the cipher. Then the message "Enter input string" appears, the string is read by the program and then changed letters according to the pattern of the cipher are displayed.
2. Write a program which is to decrypt the given string (encrypted with a simple substitution cipher):
   DOXQRIRGB MFBOTPWX LPLYX ALPQXGB MIRPFHX
4. Read from the user a string of text followed by a period character (.) Then divide the text into words and all longer than 4 characters write to a separate string table. Finally, output an inscription composed of the 2nd and 3rd characters of each element in this new table.
5. Declare a longer text containing words that are forbidden (eg, 'mathematics') and depict all instances of something neutral (eg 'flower').
6. Write a function to check if the word is a palindrome.

Excercises

7. Write a program that counts all vowels in the given string, and displays their number.

   Example:

   „Ala ma kota”

   a – 4

   o – 1

8. Write a program that reads a string of characters from the user, ended by pressing the Enter key (only characters from English alphabet are allowed). The program should display:
   - The number of white characters in the text (understood as spaces and a horizontal tab)
   - The number of letters in the text
   - The number of vowels in the text (only from the English alphabet)
   - The sum of all digits given in the text

   Example: Input text: Ala ma 2 koty i 15 rybek.

   White chars: 7

   Letters: 15

   Vowels: 8

   Sum: 8