# INFORMATYKA

STRUKTURY

## CEL LABORATORIUM

Celem zajęć jest zapoznanie uczestniczki/uczestnika z tablicami znaków i strukturami w języku C. Zajęcia te wieńczą cykl zajęć dotyczący języka C. W trakcie zajęć przedstawiony zostanie sposób deklaracji, definicji i pracy ze strukturami w języku C. Zajęcia kończą się szeregiem zadań praktycznych.

## POCZĄTEK ZAJĘĆ

Uruchom program Oracle **VM VirtualBox** i maszynę wirtualną wskazaną przez osobę prowadzącą zajęcia. Jeżeli prace wykonujesz na maszynie wirtualnej SO_BD_IO, otwórz menu „`Start`”, najedź kursorem na pozycję „`Wszystkie programy`”, a następnie na „`Microsoft Visual Studio .NET 2003`”. Możesz również skorzystać z własnego komputera lub z innych środowisk obecnych na komputerach w laboratorium. Upewnij się, że Twoje środowisko obsługuje języki C i C++.

Możesz skorzystać z projektu utworzonego na poprzednich zajęciach, jeżeli masz do niego dostęp. Możesz też założyć nowy projekt. Wybór należy do Ciebie.

## CIĄGI ZNAKÓW

Już na pierwszych zajęciach z języka C, wypisywaliśmy na ekranie pewien ciąg znaków. Do tej pory, potrafiliśmy jednak tworzyć zmienne przechowujące wyłącznie pojedyncze znaki (zmienne typu `char`). Poznawszy ideę tablic, być może domyślasz się, że żeby przechować ciąg znaków w zmiennej, należy zdefiniować tablicę znaków.

Jeżeli tak uważasz, to masz rację. Ciągi znaków są prawie zwyczajnymi tablicami znaków. Dlaczego prawie? Ponieważ oprócz samych znaków, na samym końcu takiej tablicy, musi znajdować się wartość zero (której odpowiada specjalna sekwencja: „\0”). Analizując więc ciąg znaków „*Hello world!*”, ma on w rzeczywistości przynajmniej 13 znaków:

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **'H'** | **'e'** | **'l'** | **'l'** | **'o'** | **' '** | **'w'** | **'o'** | **'r'** | **'l'** | **'d'** | **'!'** | **\0** |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |

Tablica przechowująca ciąg znaków może mieć więcej znaków niż sam ciąg. Powyższa tablica może być dowolnie duża, ale znak „`\0`” sprawia, że znaki znajdujące się dalej w tablicy, są ignorowane.

## TABLICE ZNAKÓW

Tablicę znaków deklarujemy jak każdą inną tablicę. Różnica pojawia się dopiero podczas definicji. Możemy skorzystać z klasycznej metody przypisywania znakom wartości jeden, po drugim, a możemy po prostu przypisać wartość w takiej postaci, w jakiej do tej pory podawaliśmy ją chociażby do funkcji `printf`. Tablica, podczas samej deklaracji, zawiera losowe wartości. Próba jej wypisania na ekranie, może mieć różne skutki.

# ZADANIE 1

Przepisz, skompiluj i wypróbuj działanie poniższego kodu:

```
#include <stdio.h>

int main(void)
{
    char tab1[100];
    char tab2[100] = {'t', 'e', 's', 't'};
    char tab3[100] = "Hello world!";

    printf("%s\n", tab1);
    printf("%s\n", tab2);
    printf("%s\n", tab3);
    return 0;
}
```

## FUNKCJE DO PRACY Z CIĄGAMI ZNAKÓW

W związku z tym, że praca z ciągami znaków jest jednym z najczęściej występujących zagadnień, w bibliotece standardowej języka C, znalazło się miejsce dla kilku funkcji, które pomagają pracować z tablicami znaków. Żeby z nich skorzystać, musimy dołączyć bibliotekę `string` (ang. *„string" – „ciąg znaków"*):

```
#include <string.h>
```

Najczęściej używane funkcje zebrano w tabeli poniżej. W przykładach założono, że zmienne `c1` i `c2`, są tablicami znaków:

| Funkcja | Opis | Przykład |
|---|---|---|
| `strcat` | Dodaje c2 do końca c1. Znak '\0' na końcu c1, jest nadpisywany przez pierwszy znak c2. | `strcat(c1, c2);` |
| `strcmp` | Porównuje dwa łańcuchy znaków ze sobą. Jeżeli są równe, zwraca 0. Jeżeli c1<c2, zwraca -1. Jeżeli c1>c2, zwraca 1. | `int coWieksze = 0;`<br>`coWieksze = strcmp(c1, c2);` |
| `strcpy` | Kopiuje zawartość c2 do c1. Jeżeli c1 zawierało wcześniej jakąś treść, staje się ona niedostępna. | `strcpy(c1, c2);` |
| `strlen` | Zwraca długość ciągu znaków | `int dlugosc = 0;`<br>`dlugosc = strlen(c1);` |

Wszystkie te funkcje możesz oczywiście zaimplementować samodzielnie. Jako, że ciąg znaków to tablica, możesz do poszczególnych znaków odwoływać się, jak do elementów zwykłej tablicy. Dzięki temu, funkcję `strcpy`, można zaprogramować za pomocą zwykłej pętli for i warunku sprawdzającego, kiedy natrafimy na symbol `\0`.

## STRUKTURY

Ostatnim interesującym elementem języka C, są struktury. Struktura jest złożonym typem danych, który grupuje w jednym obszarze pamięci dane **różnego** typu. Elementy struktury nazywamy **polami**. Pole, tak jak zwykła zmienna, ma swój typ i nazwę. Pole może również być typu złożonego – może więc być tablicą lub inną strukturą. Może też być wskaźnikiem.

Żeby zadeklarować strukturę, należy użyć słowa kluczowego `struct`. Po tym słowie, następuje nazwa struktury, a następnie nawias klamrowy, otwierający listę pól struktury. Pola mogą być **wyłącznie deklaracjami**, język C **nie pozwala na definiowanie pól**. Po zadeklarowaniu wszystkich pól, następuje zamykający nawias klamrowy oraz **średnik**. Nieumieszczenie średnika na końcu deklaracji struktury jest **błędem**, gdyż deklaracja struktury jest traktowana jak deklaracja każdej innej zmiennej. O deklaracji struktury myśl jak o deklaracji nowego, nieistniejącego typu zmiennych. Deklarację przykładowej struktury znajdziesz poniżej:

```
struct osoba{
    char imie[100];
    char nazwisko[100];
    int wiek;
};
```

Możliwe jest też zdefiniowanie struktury. W takim wypadku, po klamrze zamykającej deklarację, a przed średnikiem, umieszczamy **nazwę tworzonej zmiennej strukturalnej**:

```
struct prostokat{
    float a;
    float b;
} blatStolu;
```

Zmienną typu strukturalnego nazywamy **instancją** (od łac. „instantia" – „obecność"). Gdy struktura jest zadeklarowana/zdefiniowana, możemy nową instancję utworzyć poprzez użycie słowa `struct`, a następnie deklarację, jak w przypadku zwykłej zmiennej. Oczywiście zamiast znanych nam już typów, typem będzie nazwa struktury[1]:

```
struct prostokat kwadrat;
```

Dostęp do pól struktury uzyskujemy poprzez poprzedzenie ich nazwy znakiem kropki. Nazwę należy oczywiście poprzedzić nazwą zmiennej, w której pola się znajdują:

```
kwadrat.a = 10.5;
kwadrat.b = 10.5;
printf("Prostokat ma wymiary %f*%f cm", kwadrat.a, kwadrat.b);
```

Format dostępu do pól zmieni się, jeżeli zmienna nie będzie strukturą, a wskaźnikiem do struktury. W takim wypadku, zamiast zwykłej kropki, zastosujemy „strzałkę", złożoną ze znaku minus i znaku większości:

```
struct prostokat *romb = &kwadrat;
romb->a = 12.25;
romb->b = 12.25;
printf("Prostokat ma wymiary %f*%f cm", romb->a, romb->b);
```

[1] Nie mylić z nazwą zmiennej! W przypadku instrukcji „`struct prostokat kwadrat;`", nazwa struktury to `prostokąt`, a nazwa zmiennej to `kwadrat`.

## ZADANIA DO SAMODZIELNEGO WYKONANIA

# ZADANIE 2

W rozdziale „STRUKTURY” niniejszej instrukcji, przedstawiono strukturę „osoba”. Napisz program w języku C, w którym, w funkcji main, utworzone zostaną dwie instancje tej struktury.

# ZADANIE 3 Ⓟ

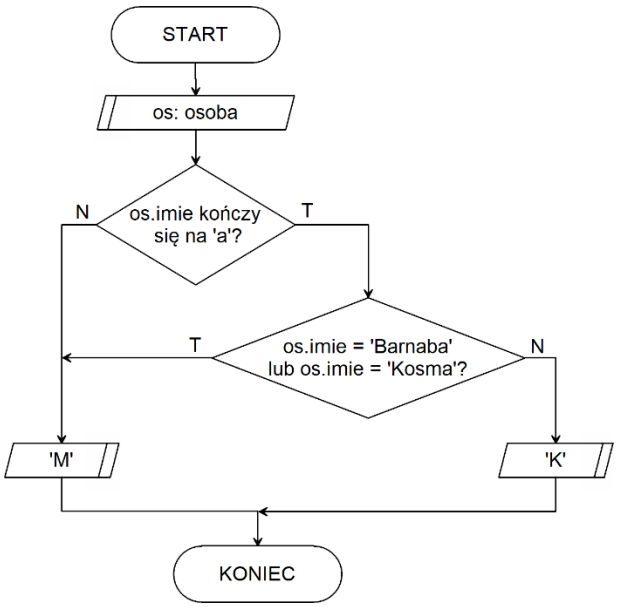
Napisz funkcję `wpiszDaneOsoby`, która przyjmuje cztery argumenty: wskaźnik do struktury typu `osoba`, dwie stuelementowe tablice znaków (`imie` i `nazwisko`) i liczbę całkowitą (`wiek`). Przypisz wartość trzech ostatnich argumentów, do odpowiednich pól zmiennej wskazywanej przez wskaźnik, będący pierwszym argumentem funkcji. Możesz skorzystać z funkcji `ctrcpy`, z biblioteki `string.h`. Wywołaj funkcję dwukrotnie jako pierwszy argument podając instancje utworzone w zadaniu 2. Imię, nazwisko i wiek, mogą być dowolne.

# ZADANIE 4 Ⓟ

Napisz funkcję `wypiszDaneOsoby`, której jedynym argumentem jest wskaźnik do struktury typu `osoba`. Funkcja powinna wypisać na ekranie tekst „[Nazwisko] [Imię] ma [X] lat”, pod „[Nazwisko]”, „[Imię]” i „[X]”, podstawiając odpowiednie wartości zawarte w strukturze przekazanej jako pierwszy argument funkcji (np. „Mickiewicz Adam ma 35 lat”). Wywołaj funkcję dwukrotnie jako argument podając instancje utworzone w zadaniu 2.

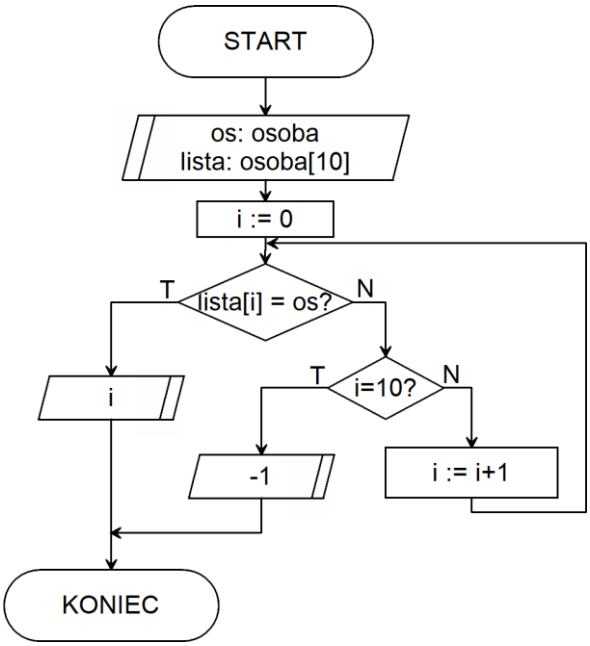
# ZADANIE 5 Ⓟ

Napisz funkcję `jakaPlec`. Zaimplementuj ją zgodnie z następującym schematem blokowym:



## ZADANIE 6 Ⓟ

Napisz funkcję `skopiujDaneOsoby`, która przyjmuje dwa argumenty, które są wskaźnikami do struktur typu `osoba`. Funkcja powinna kopiować wartość wszystkich pól instancji przekazanej jako pierwszy argument do pól instancji przekazanej jako drugi argument.

## ZADANIE 7 Ⓟ (NAGRADZANE DWOMA PLUSAMI)

Napisz funkcję `czyObecna`, która przyjmuje dwa argumenty. Pierwszy argument jest wskaźnikiem do struktur typu `osoba`, a drugi argument dziesięcioelementową tablicą struktur typu osoba. Zaimplementuj ją zgodnie z następującym schematem blokowym: