Informatika 3.
2. előadás: C alapok

Kovács Kristóf

Budapesti Műszaki és Gazdaságtudományi Egyetem

2024-02-20

Változók

- Deklarálás

```
int x;
```

Változók

- Deklarálás

```
int x;
```

- Definiálás

```
int x = 5;
```

Változók

- Deklarálás

```
int x;
```

- Definiálás

```
int x = 5;
```

- Értékadás

```
x = 5;
```

Változók

- Deklarálás

```
int x;
```

- Definiálás

```
int x = 5;
```

- Értékadás

```
x = 5;
```

- Egy változót deklarálni vagy definiálni kell mielőtt használhatnánk.

Változók

- Deklarálás

```
int x;
```

- Definiálás

```
int x = 5;
```

- Értékadás

```
x = 5;
```

- Egy változót deklarálni vagy definiálni kell mielőtt használhatnánk.
- Deklarálás után "undefined" milyen értéket tartalmaz a változó:

```
int x;
printf("%d", x);
...
1147283347
```

Függvények

- Függvény deklarálás

```
float teglalap(float a, float b);
```

Függvények

- Függvény deklarálás

```
float teglalap(float a, float b);
```

- Függvény definiálás

```
float teglalap(float a, float b) {
    return a * b;
}
```

Függvények

- Függvény deklarálás

```
float teglalap(float a, float b);
```

- Függvény definiálás

```
float teglalap(float a, float b) {
    return a * b;
}
```

- Egy függvényt akkor lehet használni ha korábban legalább deklaráltuk:

```
float teglalap(float a, float b);

int main(void) {
    printf("%f", teglalap(5, 7));
    return 0;
}
float teglalap(float a, float b) {
    return a * b;
}
```

Függvények paraméterei

- Függvények argumentumai másolódnak:

```
void hibas(float x, float y, float sum) {
    sum = x + y;
}

int main(void) {
    float a = 0.0;
    hibas(5.0, 2.0, a);
    printf("%f", a);
    return 0;
}
...
0.0
```

Beolvasó függvény

- Írjunk függvényt mely beolvas egy számot és visszaadja

Beolvasó függvény

- Írjunk függvényt mely beolvas egy számot és visszaadja:

```
int beolvas() {
    int n;
    printf("Adj meg egy egesz szamot: ");
    scanf("%d", &n);
    return n;
}

int main(void) {
    int a = beolvas();
    printf("Az %d szam negyzete: %d", a, a * a);
    return 0;
}
```

struct

- Írjunk függvényt mely beolvas két számot

struct

- Írjunk függvényt mely beolvas két számot:

```
struct ketto {
    int a;
    int b;
};
struct ketto beolvas() {
    int a,b;
    scanf("%d", &a);
    scanf("%d", &b);
    struct ketto k;
    k.a = a;
    k.b = b;
    return k;
}
```

struct

- Írjunk függvényt mely beolvas két számot:

```
struct ketto {
    int a;
    int b;
};
struct ketto beolvas() {
    struct ketto k;
    scanf("%d", &(k.a));
    scanf("%d", &(k.b));
    return k;
}
int main() {
    struct ketto s = beolvas();
    printf("%d, %d", s.a, s.b);
    return 0;
}
```

typedef

- A *typedef* kulcsszóval új nevet adhatunk típusoknak:

```
struct ketto {
    int a, b;
};
typedef struct ketto catto;
catto beolvas() {
    catto k;
    scanf("%d", &(k.a));
    scanf("%d", &(k.b));
    return k;
}
int main() {
    catto s = beolvas();
    printf("%d, %d", s.a, s.b);
    return 0;
}
```

typedef

- A *typedef* kulcsszóval új nevet adhatunk típusoknak:

```
typedef struct ketto {
    int a, b;
} catto;
catto beolvas() {
    catto k;
    scanf("%d", &(k.a));
    scanf("%d", &(k.b));
    return k;
}
int main() {
    catto s = beolvas();
    printf("%d, %d", s.a, s.b);
    return 0;
}
```

Gyakorlás

- Definiáljunk adatszerkezetet mely egy személy azonosítására szolgáló adatait tartalmazza (tegyük fel, hogy létezik a *string* típus.

Gyakorlás

- Definiáljunk adatszerkezetet mely egy személy azonosítására szolgáló adatait tartalmazza (tegyük fel, hogy létezik a *string* típus.
  - Név
  - Születési név
  - Anyja neve
  - Születésnap
  - Szülőváros

Pointerek

- Miért van memóriaszemét egy változóban ha nem adunk neki kezdő értéket?

Pointerek

- Miért van memóriaszemét egy változóban ha nem adunk neki kezdő értéket?
- Minden változó a számítógép memóriájában van tárolva.

Pointerek

- Miért van memóriaszemét egy változóban ha nem adunk neki kezdő értéket?
- Minden változó a számítógép memóriájában van tárolva.
- A memóriaszemét a memória korábban tárolt értékek egy szelete.

Pointerek

- Miért van memóriaszemét egy változóban ha nem adunk neki kezdő értéket?
- Minden változó a számítógép memóriájában van tárolva.
- A memóriaszemét a memória korábban tárolt értékek egy szelete.
- Minden változónak elkérhetjük a helyét a memóriában, a *pointer*-ét.

Pointerek

- Miért van memóriaszemét egy változóban ha nem adunk neki kezdő értéket?
- Minden változó a számítógép memóriájában van tárolva.
- A memóriaszemét a memória korábban tárolt értékek egy szelete.
- Minden változónak elkérhetjük a helyét a memóriában, a *pointer*-ét.

```
int main() {
    int x = 25;
    int *x_p = &x;
    printf("A memoriaban %p helyen tarolt ertek %d.",
      x_p, *x_p);
    return 0;
}
```

```
...
A memoriaban 0x7ffd97aeb0fc helyen tarolt ertek 25.
```

Pointerek 2

- A két pointer operátor:
  - &

```
int x = 25;
int *x_p = &x;
```

  - *

```
int x = 25;
int *x_p = &x;
int y = *x_p;
```

Pointerek 2

- A két pointer operátor:
  - &

```
int x = 25;
int *x_p = &x;
```

  - *

```
int x = 25;
int *x_p = &x;
int y = *x_p;
```

- Pointer pointerét is elkérhetjük (és így tovább):

```
int x = 25;
int *x_p = &x;
int **x_pp = &x_p;
int y = **x_pp;
```

Beolvasás pointerekkel

```c
void beolvas(int *a, int *b) {
    scanf("%d", a);
    scanf("%d", b);
}

int main() {
    int a, b;
    beolvas(&a, &b);
    printf("%d, %d", a, b);
    return 0;
}
```

Gyakorlás

- Írjunk függvényt mely kiszámolja a kapott oldalhosszúságú téglalap területét és kerületét! Kikötés, hogy a függvény *void* visszatérési értékű.

Gyakorlás

- Írjunk függvényt mely kiszámolja a kapott oldalhosszúságú téglalap területét és kerületét! Kikötés, hogy a függvény *void* visszatérési értékű.
- Írjunk függvényt mely a pointeren keresztül kapott egész számot négyzetre emeli!

Tömbök

- A tömbökre tekinthettek úgy mint limitált python listák

Tömbök

- A tömbökre tekinthettek úgy mint limitált python listák:

```
int t[3];
t[0] = 1;
t[1] = 2;
t[2] = 5;
```

Tömbök

- A tömbökre tekinthettek úgy mint limitált python listák:

```
int t[3];
t[0] = 1;
t[1] = 2;
t[2] = 5;
```

- Így is lehet, de csak inicializáláskor működik ez:

```
int t[] = {1, 2, 5};
```

Tömbök

- A tömbökre tekinthettek úgy mint limitált python listák:

```
int t[3];
t[0] = 1;
t[1] = 2;
t[2] = 5;
```

- Így is lehet, de csak inicializáláskor működik ez:

```
int t[] = {1, 2, 5};
```

- Tömböknek fix mérete van, ezt deklarálásnál meg kell adni.

Beolvasás tömbökkel

- Ez miért működik?:

```
void beolvas(int t[]) {
    scanf("%d", &t[0]);
    scanf("%d", &t[1]);
    scanf("%d", &t[2]);
}

int main() {
    int t[3];
    beolvas(t);
    printf("%d, %d, %d", t[0], t[1], t[2]);
    return 0;
}
```

Tömb mint pointer

- A tömbök pointerek, a pointerek tömbök

Tömb mint pointer

- A tömbök pointerek, a pointerek tömbök:

```
int main() {
    int t[] = {1, 2, 5};
    int x = 3;
    int *x_p = &x;
    printf("A tomb elso eleme: %d.\n", t[0]);
    printf("A tomb mint pointer tarolt erteke: %d.\n",
        *t);
    printf("A pointer mint tomb elso eleme: %d.\n",
        t[0]);
    return 0;
}
```

```
...
A tomb elso eleme: 1.
A tomb mint pointer tarolt erteke: 1.
A pointer mint tomb elso eleme: 1.
```

Tömb mint pointer 2

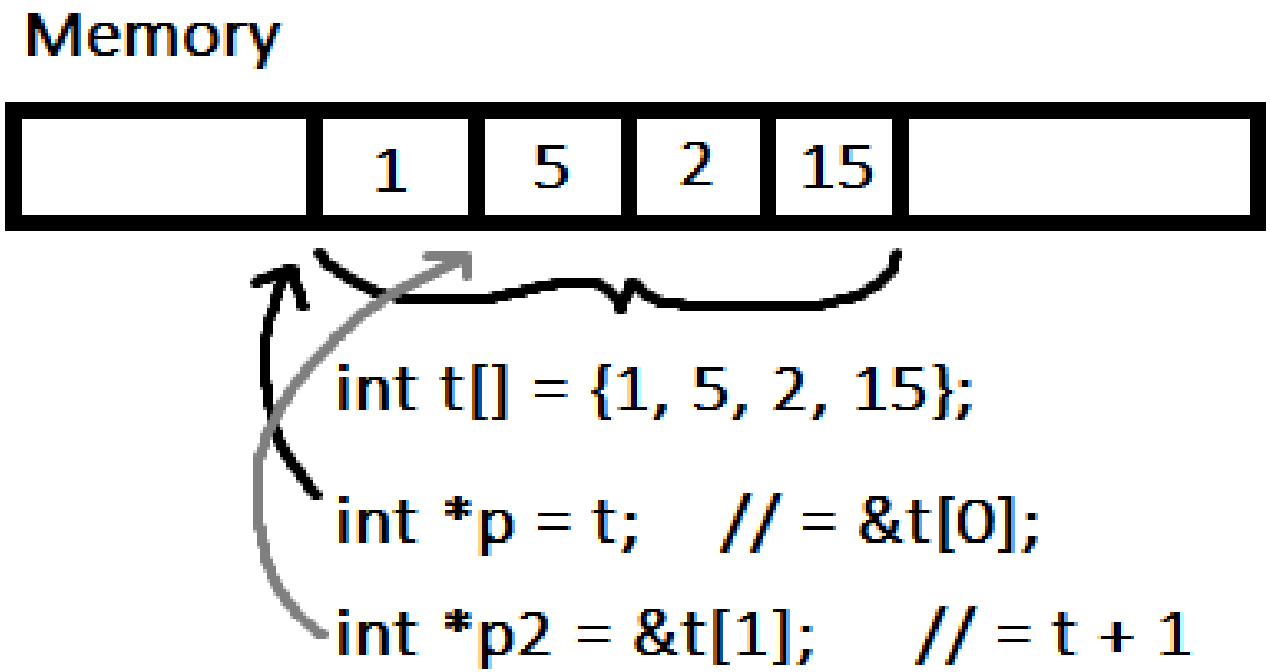
- Ha egy pointerhez egész számot adunk az olyan mintha annyival lépnénk előre a memóriában:

```
int t[] = {1, 5, 2, 15};
int *p = t + 1;
int x = *(t + 1);      // 5
```

Tömb mint pointer 2

- Ha egy pointerhez egész számot adunk az olyan mintha annyival lépnénk előre a memóriában:

```
int t[] = {1, 5, 2, 15};
int *p = t + 1;
int x = *(t + 1);      // 5
```



Gyakorlás

- Írjunk függvényt, mely a kapott tömböt megfordítja!

Gyakorlás

- Írjunk függvényt, mely a kapott tömböt megfordítja!
- Írjunk programot, mely bekér a felhasználótól egy számot és kiírja, hogy az annyiadik hónap hány napos!

Gyakorlás

- Írjunk függvényt, mely a kapott tömböt megfordítja!
- Írjunk programot, mely bekér a felhasználótól egy számot és kiírja, hogy az annyiadik hónap hány napos!
- Írjunk függvényt mely a korábban megírt négyzetre emelős függvényt alkalmazza a kapott tömb minden elemén!

kisZH kérdések

- Definiáljunk egy *float* tömböt a 2.2, 5.4, 1.4 elemekkel!
- Írjunk függvényt mely visszaadja a bemeneti egész szám négyzetét!
- Hozzunk létre egy *int* változót és egy rá mutató pointer változót!
- Definiáljunk *struct* típust mely 3 *float* változót tárol!
- Adjunk példát változó definíció és deklarációra!