Przypomnienie

Co było ostatnio?

Przypomnienie

```
//deklarowanie funkcji/zmiennych globalnych
int main(void) {
  //kod programu
  return 0;
 }
```

Przypomnienie

```
//deklarowanie funkcji/zmiennych globalnych
int main(void) {
  //kod programu
  return 0;
 }
```

- Średniki

Przypomnienie

```
//deklarowanie funkcji/zmiennych globalnych
int main(void) {
  //kod programu
  return 0;
 }
```

- Średniki
- Deklaracja zmiennych:

Przypomnienie

```
//deklarowanie funkcji/zmiennych globalnych
int main(void) {
  //kod programu
  return 0;
 }
```

- Średniki
- Deklaracja zmiennych: `int var1 = 0; int var2;`

Przypomnienie

```
//deklarowanie funkcji/zmiennych globalnych
int main(void) {
  //kod programu
  return 0;
 }
```

- Średniki
- Deklaracja zmiennych: `int var1 = 0; int var2;`
- Zmienne: **rozmiar** i **zakres**

Przypomnienie

```
//deklarowanie funkcji/zmiennych globalnych
int main(void) {
  //kod programu
  return 0;
 }
```

- Średniki
- Deklaracja zmiennych: `int var1 = 0; int var2;`
- Zmienne: **rozmiar** i **zakres**
- Definicja funkcji:

Przypomnienie

```
//deklarowanie funkcji/zmiennych globalnych
int main(void) {
  //kod programu
  return 0;
 }
```

- Średniki
- Deklaracja zmiennych: `int var1 = 0; int var2;`
- Zmienne: **rozmiar** i **zakres**
- Definicja funkcji:

```
long dodaj(long arg1, long arg2) {
  return arg1 + arg2;
}
```

Przypomnienie

```
//deklarowanie funkcji/zmiennych globalnych
int main(void) {
  //kod programu
  return 0;
 }
```

- Średniki
- Deklaracja zmiennych: `int var1 = 0; int var2;`
- Zmienne: **rozmiar** i **zakres**
- Definicja funkcji:

```
long dodaj(long arg1, long arg2) {
  return arg1 + arg2;
}
```

- Petle:

Przypomnienie

```
//deklarowanie funkcji/zmiennych globalnych
int main(void) {
  //kod programu
  return 0;
 }
```

- Średniki
- Deklaracja zmiennych: `int var1 = 0; int var2;`
- Zmienne: **rozmiar** i **zakres**
- Definicja funkcji:

```
long dodaj(long arg1, long arg2) {
  return arg1 + arg2;
}
```

- Petle: `for(;;){}`, `while(1){}`

Przypomnienie

```
//deklarowanie funkcji/zmiennych globalnych
int main(void) {
  //kod programu
  return 0;
 }
```

- Średniki
- Deklaracja zmiennych: `int var1 = 0; int var2;`
- Zmienne: **rozmiar** i **zakres**
- Definicja funkcji:

```
long dodaj(long arg1, long arg2) {
  return arg1 + arg2;
}
```

- Petle: `for(;;){}`, `while(1){}` i instrukcje:

Przypomnienie

```
//deklarowanie funkcji/zmiennych globalnych
int main(void) {
  //kod programu
  return 0;
 }
```

- Średniki
- Deklaracja zmiennych: `int var1 = 0; int var2;`
- Zmienne: **rozmiar** i **zakres**
- Definicja funkcji:

```
long dodaj(long arg1, long arg2) {
  return arg1 + arg2;
}
```

- Petle: `for(;;){}`, `while(1){}` i instrukcje: `if(5){} else {}`

- Wyswietlanie:

- Wyswietlanie: `printf("%d %c", a, b);`

- Wyswietlanie: `printf("%d %c", a, b);`
- Wczytywanie:

- Wyswietlanie: `printf("%d %c", a, b);`
- Wczytywanie: `scanf("%Ld", &c);`

- Wyswietlanie: `printf("%d %c", a, b);`
- Wczytywanie: `scanf("%Ld", &c);`
- Bloki:

- Wyswietlanie: `printf("%d %c", a, b);`
- Wczytywanie: `scanf("%Ld", &c);`
- Bloki:

```
{
  int i = 5, j = 4;
  {
    int j = 10;
    i += j;
  }
  printf("%d", i);
}
```

- Wyswietlanie: `printf("%d %c", a, b);`
- Wczytywanie: `scanf("%Ld", &c);`
- Bloki:

```
{
  int i = 5, j = 4;
  {
    int j = 10;
    i += j;
  }
  printf("%d", i);
}
```

- Bloki:

```
{
  int j = 4;
  {
    int i = 10;
    i += j;
  }
  printf("%d", i);
}
```

- Bloki:

```
{
  int i = 5, j = 4;
  {
    int i = 10;
    i += j;
  }
  int i = 10;
  printf("%d", i);
}
```

Jakiego typu *najbardziej* brakuje w C?

Jakiego typu *najbardziej* brakuje w C?
Nie ma typu string.

Tablice

Możemy definiować tablice — podobne do list w Pythonie
Dana tablica może przechowywać elementy tylko jednego typu.
Spójny obszar pamieci

Tablice

Możemy definiować tablice — podobne do list w Pythonie
Dana tablica może przechowywać elementy tylko jednego typu.
Spójny obszar pamieci

```
int tab1[100]; // 100 intów, nr. 0-99
tab1[4] = 20;
printf("%d", tab1[50]); //pamietamy o inicjalizacji!
```

Tablice

Możemy definiować tablice — podobne do list w Pythonie
Dana tablica może przechowywać elementy tylko jednego typu.
Spójny obszar pamieci

```
int tab1[100]; // 100 intów, nr. 0-99
tab1[4] = 20;
printf("%d", tab1[50]); //pamietamy o inicjalizacji!

int tab2[100] = tab1; //TO JEST ŹLE!!!
```

Tablice

Możemy definiować tablice — podobne do list w Pythonie
Dana tablica może przechowywać elementy tylko jednego typu.
Spójny obszar pamieci

```
int tab1[100]; // 100 intów, nr. 0-99
tab1[4] = 20;
printf("%d", tab1[50]); //pamietamy o inicjalizacji!

int tab2[100] = tab1; //TO JEST ŹLE!!!

long int tab2[] = {1, 2, 3, 4};
long int tab3[4] = {1, 2, 3, 4};
long int tab4[4] = {1, 2};
```

Tablice

Możemy definiować tablice — podobne do list w Pythonie
Dana tablica może przechowywać elementy tylko jednego typu.
Spójny obszar pamieci

```
int tab1[100]; // 100 intów, nr. 0-99
tab1[4] = 20;
printf("%d", tab1[50]); //pamietamy o inicjalizacji!

int tab2[100] = tab1; //TO JEST ŹLE!!!

long int tab2[] = {1, 2, 3, 4};
long int tab3[4] = {1, 2, 3, 4};
long int tab4[4] = {1, 2};

char napis[100];
scanf("%s", napis);  //nie ma &
printf("%s", napis);
```

Tablice wielowymiarowe

```
int tab1[10][10];

int tab2[2][3] = { {1, 2, 3}, {4, 5, 6} };
int tab3[2][3] = { 1, 2, 3, 4, 5, 6 }; // to jest dobrze
```

Tablice wielowymiarowe

```
int tab1[10][10];

int tab2[2][3] = { {1, 2, 3}, {4, 5, 6} };
int tab3[2][3] = { 1, 2, 3, 4, 5, 6 }; // to jest dobrze
```

Która deklaracja jest poprawna?

```
int tab2[][] = { {1, 2, 3}, {4, 5, 6} };
int tab2[2][] = { {1, 2, 3}, {4, 5, 6} };
int tab2[][3] = { {1, 2, 3}, {4, 5, 6} };
```

Tablice wielowymiarowe

```
int tab1[10][10];

int tab2[2][3] = { {1, 2, 3}, {4, 5, 6} };
int tab3[2][3] = { 1, 2, 3, 4, 5, 6 }; // to jest dobrze
```

Która deklaracja jest poprawna?

```
int tab2[][] = { {1, 2, 3}, {4, 5, 6} };
int tab2[2][] = { {1, 2, 3}, {4, 5, 6} };
int tab2[][3] = { {1, 2, 3}, {4, 5, 6} };
```

Odp: 3

Tablice wielowymiarowe

```
int tab1[10][10];

int tab2[2][3] = { {1, 2, 3}, {4, 5, 6} };
int tab3[2][3] = { 1, 2, 3, 4, 5, 6 }; // to jest dobrze
```

Która deklaracja jest poprawna?

```
int tab2[][] = { {1, 2, 3}, {4, 5, 6} };
int tab2[2][] = { {1, 2, 3}, {4, 5, 6} };
int tab2[][3] = { {1, 2, 3}, {4, 5, 6} };
```

Odp: 3

```
int tab2[10][10][10][10];
```

Tablice i funkcje

Przekazywanie tablicy do funkcji:

```
void wypisz(int tab[], int rozmiar)
{
  for (int i=0; i<rozmiar; ++i)
    printf("%d ", tab[i]);
  printf("\n");
}
```

Tablice i funkcje

Przekazywanie tablicy do funkcji:

```
void wypisz(int tab[], int rozmiar)
{
  for (int i=0; i<rozmiar; ++i)
    printf("%d ", tab[i]);
  printf("\n");
}
```

Czy mozna zwrocic tablice w funkcji?

Tablice i funkcje

Przekazywanie tablicy do funkcji:

```
void wypisz(int tab[], int rozmiar)
{
  for (int i=0; i<rozmiar; ++i)
    printf("%d ", tab[i]);
  printf("\n");
}
```

Czy mozna zwrocic tablice w funkcji?
Nie!

Typedef

```
typedef long long int moj_typ;
typedef int moj_typ_tab[10];
```

Struktury

```
struct Ksiazka
{
  char autor[50];
  char tytul[50];
  int rok_wydania;
  int liczba_stron;
};

...

struct Ksiazka potop;
struct Ksiazka ksiazki[1000];
```

Struktury + typedef

```
typedef struct Ksiazka ksiazka;
```

Struktury + typedef

```
typedef struct Ksiazka ksiazka;

typedef struct Ksiazka
{
char autor[50];
char tytul[50];
int rok_wydania;
int liczba_stron;
} ksiazka;

...

ksiazka potop;
```

Struktury + funkcje

```
typedef struct Ksiazka
{
char autor[50];
char tytul[50];
int rok_wydania;
int liczba_stron;
} ksiazka;


void wypisz_ksiazke(ksiazka k) {
printf("%s '%s' %d %d \n",
  k.autor,
  k.tytul,
  k.rok_wydania,
  k.liczba_stron);
}
```

Pytania

```
typedef struct Struktura1{
  char a;
  char b;
  char c;
} struktura1;

typedef struct Struktura2{
  char a;
  char b;
  char c;
  int x;
} struktura2;
...
struktura1 s1; struktura2 s2;
printf("%lu\n", sizeof(s1) ); // Jaki bedzie wynik?
printf("%lu\n", sizeof(s2) ); //
```

Pytania

```
typedef struct Struktura1{
  int tablica[100];
} struktura1;

typedef struct Struktura2{
  int liczba;
} struktura2;

struktura1 zwroc_nowa_stukture1() {
  struktura1 s;
  return s;
}

struktura2 zwroc_nowa_stukture2() {
  struktura2 s;
  return s;
}
```

Pytania

```
typedef struct Struktura1{
  int tablica[100];
} struktura1;

typedef struct Struktura2{
  int liczba;
} struktura2;

struktura1 zwroc_nowa_stukture1() {  //ZLE
  struktura1 s;
  return s;
}

struktura2 zwroc_nowa_stukture2() { //DOBRZE
  struktura2 s;
return s;
}
```

Pytania

```
typedef struct Struktura1{
  int tablica[100];
} struktura1;

typedef struct Struktura2{
  int liczba;
} struktura2;

struktura1 oryginal1; //zle
struktura1 kopia1=oryginal1;

struktura1 oryginal2; //dobrze
struktura1 kopia2=oryginal2;
```

Pytania

```
typedef struct Struktura2{
int liczba;
} struktura2;

typedef struct Struktura1{
  int tablica[100];
  struktura2 s1;
} struktura1;
```

Punktacja (grupa MGa)

120 punktów
50 % — 3.0
85 % — 5.0