Kilka podstawowych informacji na temat typów zmiennych w języku C. Przedstawione dane dotyczą wielkości danych typów na komputerze PC. Różnica jest taka, że na przykład typ *int* w przypadku naszej płytki będzie miał 2 bajty zamiast 4.

Int jest najpopularniejszym typem zmiennej uzywanym w C/C++. Jest to zmienna 4 bajtowa (32 bit) mogąca przechowywać ponad 4 miliardy liczb (dużo) całkowitych. Jest kilka modyfikacji inta. Short int (najczęściej) 2 bajty, long int (w skrócie int) 4bajty i long long int (w skrócie long long) 8 bajtów.

Na potrzeby programowania naszego mikroprocesora warto wspomnieć też o mniej znanych uint8_t: 8 bitowy (1 bajt) int oraz uint16_t (2 bajty). Bardzo polecam uzywanie właśnie tych zmiennych ponieważ tak jak wcześniej napisałem wielkość większości typów jest zależna od architektury procesora na którym pracujemy.

Signed/unsigned char: char jest typem 8 bitowym przechowującym znaki kodu ASCI. Ale może też reprezentować liczby. Na 8 bitach zmieści się 256 liczb, w zalezności od tego czy uzyjemy bitu znaku możemy mieć zakres (-128 do 127) lub (0 do 255), odpowiednio singed, unsigned. Domyślnie char bez przedrostka występuje jako signed char.

Deklaracje static. Jeżeli w jakiejś funkcji zadeklarujemy zmienną typu static np static int to zmienna ta nie zeruje się po każdym zakończeniu funckji.

Przykład :

```
#include <iostream>

using namespace std;
void funkcja(){
	int zmienna_intowa =0;
	static int statyczna_zmienna_intowa =0;
	zmienna_intowa += 10;
	statyczna_zmienna_intowa += 10;
	printf("%d \n",zmienna_intowa);
	printf("%d \n",statyczna_zmienna_intowa);
}
int main()
{
	for(int i = 0 ; i < 3;i++){
		printf("wywolanie funkcji nr: %d \n",i);
		funkcja();
	}

	getchar();
	return 0;
}
```

Słówko extern przed zmienną informuje kompilator, że zdefiniowana zmienna ma swoją deklaracje w innej części programu.

Operacje bitowe negacja, or, and i xor:

~01101100 = 10010011
0110 | 1010 = 1110
0110 & 1010 = 0010
0110 ^ 1010 = 1100

Operacje nie bitow, działają tak samo jak bitowe ale traktują całe wyrazenie jak jeden bit np.

!01010101 = 0
!00000000 = 1
1101 || 0101 = 1
0000 || 0000 = 0
0101 && 0001 = 1
0000 && 1111 = 0
0011 ^^ 1111 = 0
0000 ^^ 1100 = 1

Przykład w C++;

```
#include <iostream>

using namespace std;

void itob(unsigned a){
  for(unsigned m = 256; m > 0; m >>= 1)
        cout << ((a & m) ? "1" : "0");
  cout<<endl;
}

int main()
{
int temp = 13;
int temp2 = 9;
int temp3 = 0;

itob(temp);
itob(temp2);
temp3 = temp|temp2;
       itob(temp3);
temp3 = temp&temp2;
       itob(temp3);
temp3 = temp^temp2;
       itob(temp3);
temp3 = ~temp;
       itob(temp3);

       getchar();
       return 0;
}
```

Zapis *=, += etc. Jest to skrótowy sposób zapisu operacji na uzywanej zmiennej. Np.
Zammiast :
Int temp =9;
temp = temp + 1;
cout<<temp; => 10
można napisać: temp += 1;

Uzywać dzisiaj będziemy dwóch portów wejściowo/wyjściowych B i D. Obydwa można ustawić jako wejście lub wyjście. Ustawiamy je jako wejście ustawiając stan niski w rejestrze DDRx (x:= B/D) włączając wewnętrzne rezystory podciągające. Analogicznie, ustawienie jako wyjście następuje poprzez ustawienie w rejestrze DDRx stanu wyokiego. Portu B będziemy uzywać jako wyście (diody) a portu D jako wejście (przyciski) więc:
DDRB = 0xFF;
DDRD = 0x00;

Jeżeli chcemy ustawić pojedynczy bit wybranego rejsetru na 1:
DDRB |= (1<<x); //x ={0,7}
Na 0:
DDRB &=~(1<<x);
Warto sobie rozpisać jeden albo dwa przykłady ponieważ na pierwszy rzut oka nie widać że powyższy sposób działa.
Przykład do wgrania na płytke:
W tym przykładzie użyłem ustawiania pojedynczych bitów na rejestrze PORB.

```
#define F_CPU 16000000UL
#include <util/delay.h>
#include <avr/io.h>

int main(void) {
    DDRB = 0xFF;
    PORTB = 0x00;
    uint8_t temp = 0x00;
    PORTB = 35;
    while(1)
    {
        _delay_ms(1000);
        PORTB |=(1<<PA3);
        _delay_ms(1000);
        PORTB &=~(1<<PA3);
    }
}
```

Aktualny stan na nóżce portu odczytujemy z rejestru PINx.
WAR ktory prz to ktory.txt